Dimensional Modeling and E-R Modeling In The Data Warehouse

By

Joseph M. Firestone, Ph.D.

White Paper No. Eight

June 22, 1998

Introduction

Dimensional Modeling (DM) is a favorite modeling technique in data warehousing. In DM, a model of tables and relations is constituted with the purpose of optimizing decision support query performance in relational databases, relative to a measurement or set of measurements of the outcome(s) of the business process being modeled. In contrast, conventional E-R models are constituted to (a) remove redundancy in the data model, (b) facilitate retrieval of individual records having certain critical identifiers, and (c) therefore, optimize On-line Transaction Processing (OLTP) performance.

Practitioners of DM have approached developing a logical data model by selecting the business process to be modeled and then deciding what each individual low level record in the "fact table" (the grain of the fact table) will mean. The fact table is the focus of dimensional analysis. It is the table *dimensional queries segment* in the process of producing solution sets. The criteria for segmentation are contained in one or more "dimension tables" whose single part primary keys become foreign keys of the related fact table in DM designs. The foreign keys in a related fact table constitute a multi-part primary key for that fact table, which, in turn, expresses a many-to-many relationship. [1]

In a DM further, the grain of the fact table is usually a quantitative measurement of the outcome of the business process being analyzed. While the dimension tables are generally composed of attributes measured on some discrete category scale that describe, qualify, locate, or constrain the fact table quantitative measurements.

Since a dimensional model is visually represented as a fact table surrounded by dimension tables, it is frequently called a star schema. Figure One is an illustration of a DM/star schema using a student academic fact database.

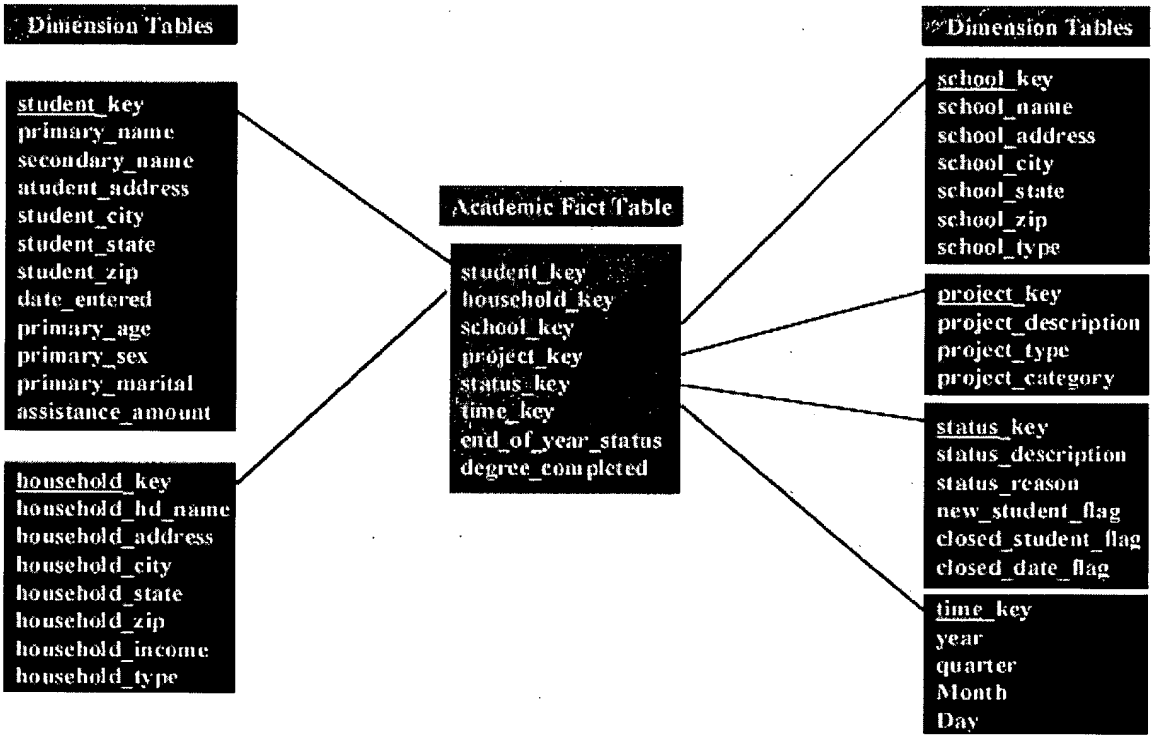


Figure One -- A Dimensional Model Star Schema of A Student Academic Fact Database

While there is consensus in the field of data warehousing on the desirability of using DM/star schemas in developing data marts, there is an on-going controversy over the form of the data model to be used in the data warehouse. The "Inmonites," support a position identified with Bill Inmon, and contend that the data warehouse should be developed using an E-R model. The "Kimballites" believe in Ralph Kimball's view that the data warehouse should always be modeled using a DM/star schema. Indeed Kimball has stated that while DM/star schemas have the advantages of greater understandability and superior performance relative to E-R models, their use involves no loss of information, because any E-R model can be represented as a set of DM/star schema models without loss of information.

In this paper I will comment on two issues related to the controversy. First, the claim that any E-R model can be represented as an equivalent set of DM/star schema models [2], and second, the question of whether an E-R structured data warehouse, absent associative entities, i.e. fact tables, is a viable concept, given recent developments in data warehousing.

Can DM Models Represent E-R Models?

In a narrow technical sense, not every E-R model can be represented as a star schema or

closely related dimensional model. It depends on the relationships in the conceptual model formalized by the logical data model.

As Ralph Kimball has pointed out on numerous occasions, star schemas represent many-to-many relationships. If there are no many-to-many relationships in an underlying conceptual model, there is no opportunity to define a series of dimensional models. That is, the possibility of a dimensional model is associated with the presence of many-many relationships of whatever order. On the other hand, an E-R model can be defined whether or not many-many relationships exist. But without them it would have no fact tables.

Having said the above, it really doesn't directly address the central question of whether an *E-R data warehouse* model can always be represented as a series of dimensional models. But it does shed some light on it. Specifically, the answer to the question depends on whether the underlying conceptual model of a data warehouse must always contain many-to-many relationships. I think the answer to this question is yes, and that it follows that an E-R data warehouse can be expressed as a star schema. Here are my reasons.

(1) Data warehouses must contain "grain" attributes in the sense of the term specified by Ralph Kimball in The Data Warehouse Toolkit. This is a necessary conclusion for anyone who believes either in a queryable data warehouse, or in a data warehouse that will primarily serve as a feeder system for queryable data marts. In either case, the grain attributes must be available as part of the data warehouse, because they provide data on the extent to which any business is meeting its goals or objectives. Without such attributes, business performance can't be evaluated, and a primary DSS-related purpose of the data warehouse architecture can't be fulfilled.

(2) If the grain attributes are present in the data warehouse, what kinds of relationships will be associated with them and what kinds of entities will contain them? In the underlying conceptual model of the data warehouse, there will be attributes that are causally related to the grain attributes, attributes that are effects of the grain attributes, and attributes such as product color, geographic level, and time period that are descriptive of the grain attributes. In the conceptual model, the grain attributes will be associated with many-many relations among these different classes of factors. How can these many-many relations be resolved in a formal model, whether E-R or dimensional?

(3) The various causal, effect, and descriptive factors will be contained in fundamental entities, and perhaps in attributive entities, or sub-type entities as well. In a correct E-R or dimensional model, however, the entities containing the grain attributes can only be associative entities, because the grain attributes will not belong to any one fundamental entity in the model; but will be properties of a many-many relation (an n-ary association) among fundamental entities.

Since fact tables are resolved many-many relations among fundamental entities, it follows that in a correct E-R model, fact tables are a necessary consequence of grain attributes and of

standard E-R modeling rules requiring conceptual correctness and conceptual and syntactic completeness. It goes without saying that fact tables are also the means of resolving many-many relationships in dimensional models.

(4) If fact tables must be present in correct E-R models, it still doesn't follow, however, that the fundamental entities related to them must be de-normalized dimension tables as specified in dimensional models. Here, in my view, is where the major distinction between dimensional and E-R data warehouse models will be found.

In E-R models, normalization through addition of attributive and sub-type entities destroys the clean dimensional structure of star schemas and creates "snowflakes," which, in general, slow browsing performance. But in star schemas, browsing performance is protected by restricting the formal model to associative and fundamental entities, unless certain special conditions (pointed out in "Toolkit," and in Ralph Kimball's various DBMS columns) exist.

So, that's it. In data warehouses, conventional E-R models and Star Schemas are both options, and this is due to the semantics of data warehouses as DSS applications requiring many-to-many relationships containing essential grain attributes. Kimball's position is therefore essentially correct: a data warehouse E-R model can be represented as a series of dimensional models. But this argument has an additional implication I'd like to see widely discussed.

I emphasized earlier that both correct dimensional and E-R models rely on fact tables to resolve the many-many relations encompassing grain attributes that are so essential for the data warehouse. If this is true, then why are fact tables so frequently associated with dimensional data warehouse models and not with correct E-R data warehouse models? I suspect this may be because many E-R data warehouse models may not always explicitly recognize many-many relations and the need to resolve them with associative entities, i.e. fact tables. Instead, these models are being defined with fundamental entities containing some of the characteristics of associative entities but also carrying with them the risks of confusion, contradiction, and redundancy inherent in an incomplete resolution of many-to-many relationships, and ad hoc de-normalization of fundamental entities.

I can't prove that this hunch of mine is valid, and that the problem in E-R data modeling I've inferred is widespread. But there are examples of the problem in the data warehousing literature. One good example is in the recent book by Silverston, Inmon, and Graziano (Wiley, 1997) [3], called "The Data Model Resource Book." Figure 10.2 on P. 266 presents a sample data warehouse data model. This data model contains no fact tables, but three tables come closest:

CUSTOMER_INVOICES,

PURCHASE_INVOICES, and

BUDGET_DETAILS.

Let's focus on CUSTOMER_INVOICES, which is typical of the three. The multi-part primary key is composed of:

INVOICE_ID, and

LINE_ITEM_SEQ.

A number of foreign keys are included as mandatory attributes, but constitute no part of the primary key, and are not determined by it. These are:

CUSTOMER_ID,

SALES_REP_ID, and

PRODUCT_CODE.

Other mandatory attributes are:

INVOICE_DATE, BILL_TO_ADDRESS_ID,

MANAGER_REP_ID, ORGANIZATON_ID,

ORG_ADDRESS_ID, QUANTITY, UNIT_PRICE,

AMOUNT, and

LOAD_DATE.

An optional attribute is PRODUCT_COST.

I believe that this entity diverges as much as it does from a fact table in a dimensional model, not because it is an E-R model-based entity, but because: (a) it fails to adequately model the conceptual distinction between customer invoice and customer sales, (b) doesn't recognize that unit price, amount, and quantity are attributes of a sale, related not only to an invoice but also to Sales Reps, Products, and Customers, and (c) in consequence doesn't correctly resolve the many-many relationship of Sales Reps, Customer Invoices, Products, and Customers. In short, the CUSTOMER_INVOICES entity, as constructed in the example, represents an error in the E-R model. That is why the QUANTITY, UNIT_PRICE, and AMOUNT attributes are not contained in a CUSTOMER_SALES associative entity, a true fact table, with a multi-part key drawn from SALES_REPS, CUSTOMER_INVOICES, PRODUCTS, and CUSTOMERS.

This point is emphasized further by looking at the star schema design for sales analysis provided in Figure 11.1 on P. 271. This design is supposed to provide an example of a departmental specific data warehouse, (or data mart). While this figure includes a CUSTOMER_SALES table that looks a lot like a fact table, it still reflects the conceptual confusion in the underlying model. Specifically, the multi-part key of this "fact table" includes INVOICE_ID, and LINE_ITEM_SEQ, as parts of the primary key. But neither attribute comes from a dimension table, nor are they degenerate dimension attributes since they are part of the primary key.

Instead they originate in the "fact table." And since from the previous CUSTOMER_INVOICES entity we know that INVOICE_ID, and LINE_ITEM_SEQ constitute a unique primary key, it follows that CUSTOMER_SALES is not an associative entity or fact table at all, but instead is another fundamental entity, very similar to CUSTOMER_INVOICES, that again confuses the distinction between CUSTOMER_INVOICES and CUSTOMER_SALES.

In short, Figure 11.1 is not a valid star schema design, as Figure 10.2 is not a valid E-R model. Because neither the CUSTOMER_INVOICES entity in one, nor the CUSTOMER_SALES entity in the other, is an appropriately normalized entity, whose non-key attributes are fully dependent on the primary key. If they were, they would present properly constructed associative entities resolving many-many relations including

CUSTOMER_INVOICES, and CUSTOMER_SALES.

Again, how typical this example is of E-R modeling in data warehousing I can't say. That's the question I'd like to see more widely discussed. Is the widely perceived divergence between E-R and dimensional modeling in data warehousing due to the fact that dimensional modeling necessarily involves fact tables and E-R modeling normally does not, or is the perceived divergence due to the fact that E-R modeling practices in data warehousing are not faithful to E-R modeling principles; and if they were they would involve fact tables to exactly the same extent as dimensional models?

Is An E-R Data Warehouse Model With No Fact Tables A Viable Concept?

DM/Star schemas represent n-ary associations. N-ary associations are embodied in many-to-many relations. These may be resolved within a data model in an entity associating two or more entities. A star schema with one fact table (the associative entity) and two dimension tables represents a binary association. One with one fact table, and three dimension tables represents a ternary association, and so on.

As we have seen E-R models can also represent n-ary associations. They differ from star schemas not in the presence of fact tables, but in the fact that their dimension tables are "snowflaked" to meet the requirements of normalization.

Since star schemas and "snowflaked" E-R models represent n-ary associations, to say that another type of E-R model eliminating fact tables should be used to structure the data in the data warehouse is also to say that n-ary associations should not be used for this purpose. But n-ary associations are essential for analysis in the context of DBMS DSS applications, because analytical DSS queries employ many-to-many relationships and are frequently multi-stage in character. Many-to-many relationships can only be resolved in data models into (1) n-ary associations of various types with associative entities (fact tables), or (2) more atomic data dependency relationships in E-R models without fact tables. I think the second alternative ensures poor query response performance in large databases, and therefore discourages and often prevents execution of a multi-stage analysis process.

It does so because it provides no structure for navigating the logic of the particular n-ary association implied by an analytical DSS query, and therefore requires that the DBMS engine construct the association "on the fly." In contrast, the first alternative provides a navigational structure for such a query, with consequent good query performance, and practical implementing of multistage analysis processes. Among associative models however, a DM/Star design generally provides better navigation and performance than an E-R /Snowflake (in the absence of tools with special capability to handle the more complex snowflake model).

If one accepts this argument (and if it's correct, 95% of it is in some way owed to Ralph Kimball, and if it's wrong, the correct 95% of it is still owed to Ralph Kimball); then the claim that dimensional modeling or "snowflaked" E-R models should not be employed in the data warehouse, largely amounts to the claim that only the limited, constrained analysis supported by data dependency models without associative entities should be employed. That is, the data warehouse becomes no more than a big staging area for data marts, and has no independent analytical function of its own. I can't subscribe to this conclusion.

After all, in recent data warehousing/data mart system architectures, we've added an Operational Data Store (ODS) [4], distinct from the data warehouse, and a non-queryable centralized staging area for storing, extracted, cleansed, and transformed data and for gathering centralized metadata for implementing an Enterprise Data Mart Architecture (EDMA) [5]. Why then do we need yet another non-queryable staging area? Also, if the data warehouse is only a staging area and we can do analysis only in data marts, where do we go for enterprise-wide DSS?

Conclusion

In the context of the "Inmonite"/"Kimballite" dispute over the proper form of data warehouse data models, this paper examined: (1) the claim that any E-R model can be represented as an equivalent set of DM/star schema models; and (2) the question of whether an E-R structured data warehouse, absent associative entities, i.e. fact tables, is a viable concept given recent developments in data warehousing. A number of conclusions are supported by the arguments.

- Not every E-R model can be represented as a set of star schemas containing equivalent information;
- But every properly constructed E-R data warehousing model can be so represented;
- Many E-R data warehouse models are not properly constructed in that they don't explicitly recognize many-many relations and the need to resolve them with associative entities, i.e. fact tables.
- To use data warehousing E-R models specifying atomic data dependency relationships without fact tables is to ensure poor query response performance in large databases, and therefore discourage, and often prevent, execution of a multi-stage analysis process. In effect, it is to make the data warehouse no more than a big staging area for data marts, with no independent analytical function of its own.
- Given the development of ODSs and non-queryable centralized staging areas for storing, extracted, cleansed, and transformed data and for gathering centralized metadata for implementing an Enterprise Data Mart Architecture (EDMA); we don't need another non-queryable staging area called a data warehouse. What we do need, instead, is a dimensionally modeled data warehouse for enterprisewide DSS, prepared to provide the best in query response performance and to support the most advanced OLAP [6] functionality we can devise.

References

[1] Ralph Kimball, The Data Warehouse Toolkit (New York, NY: John Wiley & Sons, Inc., 1996), Pp. 15-16

[2] I thank Ralph Kimball for prodding myself and other participants in the dwlist@datawarehousing.com list server group about the importance of examining this issue.

[3] W. H. Inmon, Claudia Imhoff, and Ryan Sousa, Corporate Information Factory (New York, NY: John Wiley & Sons, Inc., 1998), Pp. 87-100

[4] Len Silverston, W. H. Inmon, and Kent Graziano, The Data Model Resource Book (New York, NY: John Wiley & Sons, Inc., 1997)

[5] Douglas Hackney, Understanding and Implementing Successful Data Marts (Reading, MA: Addison-Wesley, 1997), Pp. 52-54, 183-84, 257, 307-309

[6] "What is OLAP?" The OLAP Report, revised February 19, 1998, @http://www.olapreport.com/fasmi.htm

Biography

Joseph M. Firestone is an independent Information Technology consultant working in the areas of Decision Support (especially Data Marts and Data Mining), Business Process Reengineering and Database Marketing. He formulated and is developing the idea of Market Systems Reengineering (MSR). In addition, he is developing an integrated data mining approach incorporating a fair comparison methodology for evaluating data mining results. Finally, he is formulating the concept of Distributed Knowledge Management Systems (DKMS) as an organizing framework for the next business "killer app." You can e-mail Joe at eisai@home.com.

[Up] [Data Warehouses and Data Marts: New Definitions and New Conceptions]
[Is Data Staging Relational: A Comment]
[DKMA and The Data Warehouse Bus Architecture]
[The Corporate Information Factory or the Corporate Knowledge Factory]
[Architectural Evolution in Data Warehousing]
[Dimensional Modeling and E-R Modeling in the Data Warehouse]
[Dimensional Object Modeling] [Evaluating OLAP Alternatives]
[Data Mining and KDD: A Shifting Mosaic]
[Data Warehouses and Data Marts: A Dynamic View]
[A Systems Approach to Dimensional Modeling in Data Marts]
[Object-Oriented Data Warehousing]

Executive Information Systems, Inc.

Evaluating OLAP Alternatives

By

Joseph M. Firestone, Ph.D.

White Paper No. Four

March 28, 1997

OLAP Alternatives

The rush to develop data warehouses and data marts has gained considerable momentum from the presence of server-based On-line Analytical Processing (OLAP) [1] tools, including: Multidimensional server-based (MDOLAP) tools; a number of Relational OLAP (or ROLAP) alternatives; and a new tool called Sybase IQ which uses a technology we can call Vertical Technology OLAP (VTOLAP). By now the gold rush in the OLAP marketplace is in full swing, with different products striving for both marginal differentiation and substantial technical and operational advantages over competitors.

How do we choose an OLAP product for a data warehouse? Carefully, as the old saying goes, and with respect to relevant criteria. This White Paper will provide a set of criteria for product evaluation in specific project contexts. My perspective is that of an independent consultant with no product axe to grind, and no conscious a priori preference for any of the three types of OLAP, an OLAP agnostic if you will. Before getting to the criteria, however, it is useful to briefly review the three categories of OLAP and to identify some of the more prominent products in each category.

MDOLAP Alternatives

There are four leading vendors of MDOLAP servers: Arbor Software (Essbase), Kenan Technologies (Acumate Enterprise), Oracle/IRI (Express), and D & B/Pilot Software (Lightship). In addition, a server version of Cognos PowerPlay, and a new multidimensional server from the SAS Institute have recently been released.

These and other multidimensional server tools are based on the idea that a multidimensional view of business or program data is consistent with and reflects business logic and common sense, and is much more relevant to practical decisionmaking than the table-oriented view

presented by standard relational and flat file data bases. A multidimensional view of data as cubes and/or hypercubes, naturally segments data into cells lying at the intersection of causally or descriptively relevant categorical dimensions.

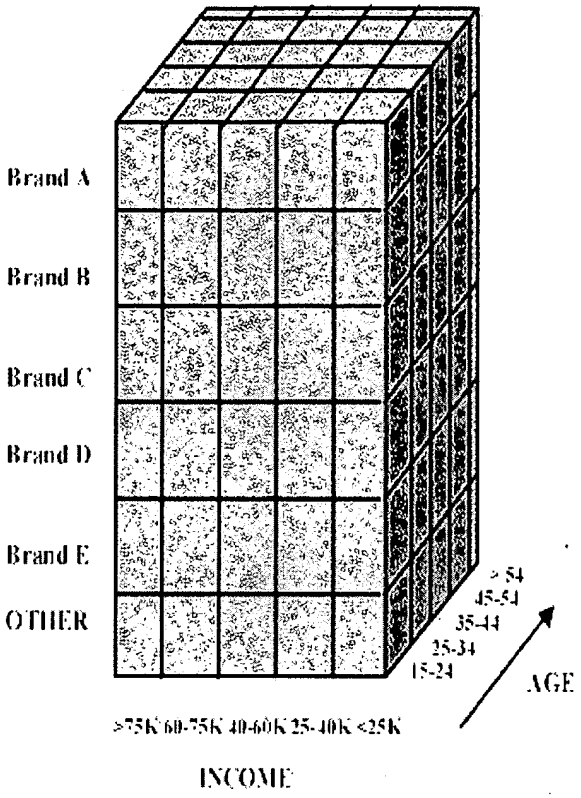


Figure One -- A Three Dimensional, Unpopulated, Array

Multidimensional database servers group field category values into dimensions, and then dimensions into multidimensional arrays. Specific field category values that may occur in data (logically possible category values) identify either the rows or columns of array dimensions, and grouped field categories identify the row or column array dimensions themselves. Figure One provides an example of a three dimensional, but unpopulated array.

Arrays are populated by placing actual field values in the cells of the array, data generally imported from flat files or relational databases. The dimensions of the array categorize the values in the cells, which, in turn, provide data variable values for the segments of the entity population defined by the categorization. The values in the cells are never values of the dimensional categories. Instead they are values of attributes that vary across the dimensions. Figure Two provides an example of a three dimensional array populated with units sold values.

Units Sold (in 000's) Varies Across the Three Dimensions of Age, Income, and Brand. The Numbers Shown Summarize Age Variations in Units Sold Across Brand, and Income. For Example, 19.1 Thousand Units of Brand A Were Sold to Individuals of All Ages with Income Greater than 75K.

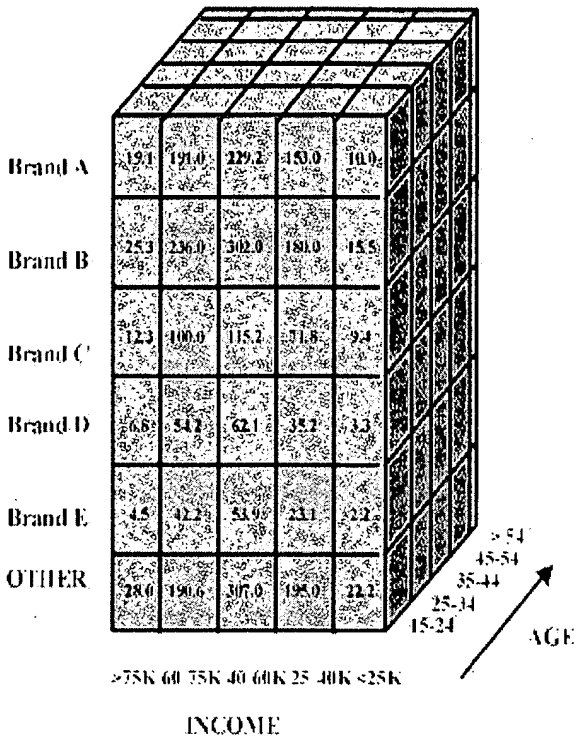


Figure Two -- A Three Dimensional Array Populated With Units Sold Values

Attributes or variables often used to populate arrays are business measurements, such as prices, costs, sales, profits, probabilities of response to promotions, and averages of customer lifetime value. In general, values in arrays are values of continuous, or at least frequency variables, while dimensional category values are those of discrete variables, or continuous variables whose values have been categorized to accomodate the OLAP perspective.

The multidimensional array has a position in the multidimensional logical model analogous to the position of the table of field values in the relational model. Arrays provide a basic segmenting structure within which data description and analysis takes place, but to perform such analysis through query processing, it is necessary to specify higher level logical operations between components of multidimensional arrays and between multidimensional arrays themselves. Multidimensional databases can call on the full set of logical operations available to relational databases.

In addition, however, there are certain logical relations and operations that are much easier to perform in multidimensional databases because they are "hard-wired" into the design of commercial products and need not be assembled from clever and exacting manipulation of more basic logical operations. These operations include:

- defining parent-child relations between dimensions and constructing dimensional hierarchies across geography, organization, time and other important organizing concepts;
- easily performing matrix calculations that allow whole vectors or slices of

arrays to be operated on at once;
- ranging or subsetting (also known as "dicing,") multidimensional arrays to provide more focused descriptions, reports, and analyses;
- rotation (also known as "data slicing,") to examine a different view of the multidimensional array being queried without having to reassemble the array from basic data; and
- aggregating or disaggregating multidimensional arrays to diplay higher or lower levels in a dimensional hierarchy such as time period, geography, or organization (known as "rolling-up" or "drilling-down").

Multidimensional databases share with relational databases either friendly query languages or visual tools that make ad hoc querying practical for database marketing analysts, sales analysts, financial ananlysts, and other practitioners of chain querying. Multidimensional database vendors, moreover, feel that their query languages are more efficient than SQL. Many examples exist comparing SQL queries with multidimensional ones. Almost invariably multidimensional queries are a fraction of the size of SQL queries. An important factor in ease of querying is the presence of the high level operations allowing data dicing, slicing, rolling-up, drilling-down and matrix arithmetic. It is easy and highly intuitive to formulate ad hoc queries using multidimensional products, and excellent visual tools exist to aid the process of ad hoc query chaining.

In the area of query performance, multidimensional databases, unaided by indexing or special hardware, exhibit improved performance over relational databases based on E-R data models. The reason for increased performance is that multidimensional databases use the high level logical operations named earlier to either retrieve summaries or counts that immediately fulfill queries, or at a minmum, to access a place in the multidimensional database that allows query processing to proceed through scanning only a small portion of the data.

An ad hoc query like "How many blue, minivans were sold by Chevy dealers in Minnesota in 1993?" requires no scanning of a multidimensional database. The cell with the answer can be reached by drilling down to State, and dicing and slicing to the face and cell of the array which has the information in it. And if the next query is, and "how many were sold in Minneapolis/St. Paul?" the answer is even faster in coming because it only requires a "drill-down."

ROLAP Alternatives

While ROLAP vendors have entered the market only in the last few years, there are four of them that have received a good bit of attention in the trade press. Two are among the fastest growing corporations in the United States. The four companies are: Microstrategy [2], Information Advantage, Stanford Technology Group (recently acquired by Informix), and IQ Software. All offer ROLAP suites including analytical server engines, reporting and analysis tools, system design tools, and web enabling software.

ROLAP products are said to produce performance comparable to that of MDOLAP tools, but

to be much more scalable to larger database sizes. ROLAP vendors adhere to the OLAP doctrine that end users should be presented with a multidimensional view of business data. But they strongly disagree that it is necessary to physically store data multidimensionally, or to use a logical model that views data as a cube or hypercube, in order to provide a multidimensional view of the data. Instead, ROLAP practitioners have developed the concepts of dimensional modeling including such data modeling concepts as the star schema, the snowflake schema, the constellation, and qualities. Figure Three illustrates a star schema with a fact table in the middle of the model and dimensional tables clustered around it.

Dimensional models clearly present multidimensional views of data. They do so by presenting a flattened view of the dimensional slices of a Decision Support system (DSS). Like a multidimensional server, ROLAP requires a separate server engine for analytical processing. But ROLAP differs from OLAP in that data remains resident on a relational database server, that participates along with the ROLAP server in a true interactive three-tier architecture. "A ROLAP engine is a set of metadata-driven software components that generate SQL -- statements and perform formulaic calculations based on users' multidimensional requests all at run time." [3]

ROLAP vendors feel that use of the relational model for multidimensional analysis is superior to use of a mutidimensional logical model. They point out that ROLAP products are much more open to standard relational OLTP architecture, and much more accessible to front-end application development, query, and reporting tools. And in a nutshell, they believe that ROLAP is just as capable as MDOLAP at providing high performance sophisticated analyses for users, while at the same time providing much

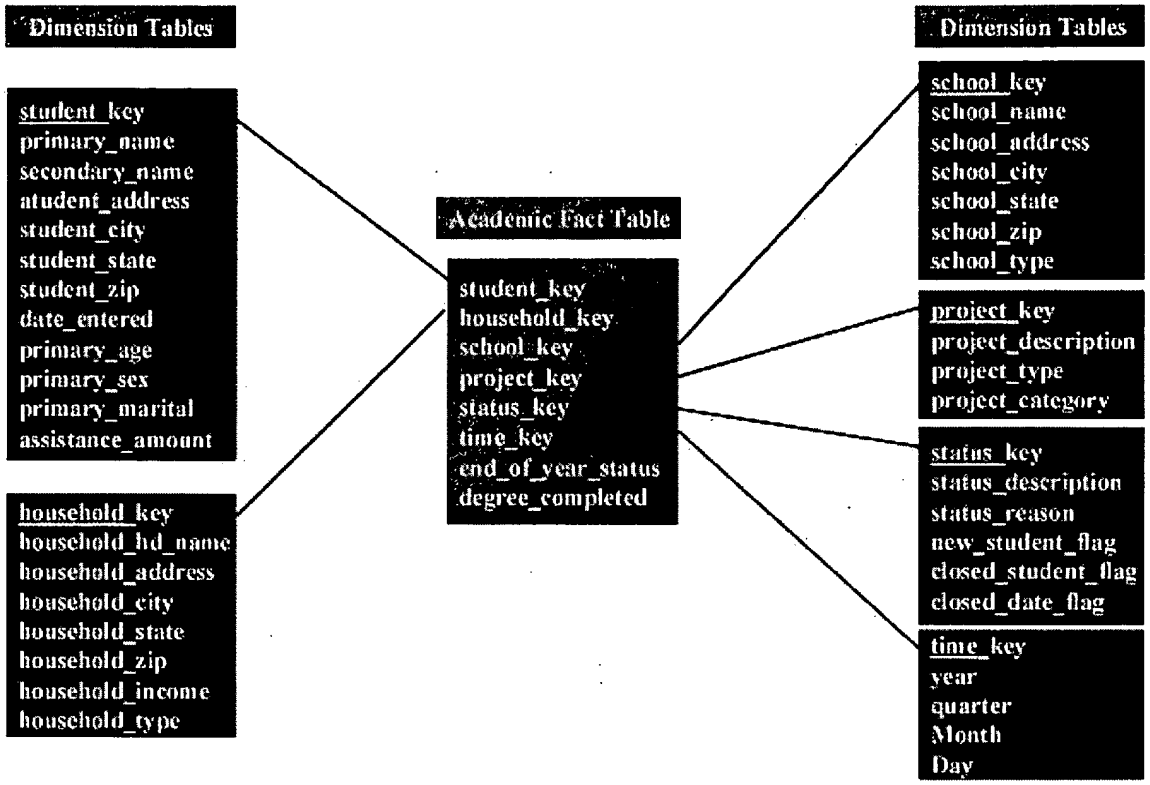


Figure Three -- A Dimensional Model Star Schema of A Student Academic Fact Database

greater scalability in relation to the number of dimensions and the amount of atomic data that enter an analysis.

Finally, ROLAP advocates believe that ROLAP is the only database technology that addresses the need for decision support in Very Large Database (VLDB) environments. In their view, ROLAP can provide unlimited scalability of dimensions and atomic data in the long run, because it is the only OLAP framework that can take advantage of parallel processing, bit-mapped indexing, and star join technologies, as well as further developments in the SQL paradigm.

VTOLAP --The Sybase IQ Alternative

Sybase IQ is based on different assumptions than either MDOLAP or ROLAP solutions to decision support systems or data warehouses. Since decision support queries often seek answers to general questions, they are oriented more to describing relations among attributes, fields, or columns of a database, rather than toward answering questions about individual or small groups of records. DSS processing is more column oriented than it is row or record oriented.

Sybase IQ begins with the assumption that such applications must be supported by data that is stored vertically, by column, rather than horizontally, by record. If data is stored vertically, rather than horizontally, it is possible to access a specific column named in a query directly, and

since only the relevant attribute is involved, to retrieve data with a minimum of disk I/O. Sybase IQ combines such column-based physical organization with advanced compression techniques, flexibility in assigning I/O block sizes to queries, prejoin indexes, B-tree indexes, and advanced bit-mapped indexing that makes bit-maps useful even at high column cardinality values of 1,000 values or more, to produce a read only decision support engine that has been benchmarked at 10 to 100 times the query performance of conventional RDBMS.

Sybase IQ also has the advantage of small storage requirements. A data mart implemented in IQ may supplement an Oracle or Sybase back-end with a data mart requiring only 50% additonal space for the IQ transformed data mart. This is in marked contrast to ROLAP designs, which may take 50 or 100 times the space of the initial relational database on which they are based.

Even though Sybase IQ is a new release for Sybase, and is revised in its present form, it is not an entirely new product. A precursor to IQ called Expressway, was released in January of 1993 by Henco software, then an 18 year old software company that had specialized in document management and text retrieval. Expressway was well on its way to becoming quite successful, with notable applications at National Liberty and MCI, when Sybase acquired it in 1995. Expressway was submerged in Sybase for awhile, but after revisions and one hopes, considerable enhancement, was re-released by Sybase as IQ. Industry Press on IQ has been good both in its previous and present incarnations. [4]

The disadvantage of IQ is that it is based, like MDOLAP servers, on a nonrelational data structure, and is a proprietary product. On the other hand, IQ does interface with both Oracle and Cognos Impromptu, as well as other open relational products, while all ROLAP server-based engines that work along with relational servers, are also proprietary. As a practical matter, there may be no greater commitment to a proprietary product in selecting the VTOLAP IQ server than there is in selecting one of its MDOLAP or ROLAP competitors.

OLAP Evaluational Criteria

Most of the following evaluation criteria are meant to apply across product classes, though a few will be useful primarily for comparisons within a product class. The criteria are unweighted. To carry out a quantitative comparative evaluation, the criteria need to be structured further and to be prioritized through use of a formal method for setting priorities in the context of a specific application.

The method I like best for this kind of task is Saaty's Analytic Hierarchy Process (AHP). you can learn about it and its myriad uses from the Expert Choice Internet site [5], and you may also find Zahedi's application of AHP to the task of software evaluation [6] to be instructive in the present context.

- **Database size capacity of product relative to your data warehouse or data mart size requirements**

MDOLAP tools have been rapidly increasing their capacity to handle large databases as vendor competition has increased; but they generally have less capacity to handle sheer volume than ROLAP tools, or Sybase IQ. But the real question is: does the tool being evaluated have enough capacity for your DSS application?

- **Ability of tool to scale to the number of dimensions required by your DSS application.**

To evaluate this, you need to know what dimensions are required, and therefore to do a requirements analysis. In another White Paper, I developed a Systems Approach to Dimensional Data Modeling [7]. While that approach is specific to ROLAP development, the first three steps of the approach apply equally well to other types of OLAP development. The systems approach describes the steps to be followed in arriving at basic measurements tracked in OLAP applications, and dimensions that will be used to view and segment the units of analysis whose measurements are being tracked, into groups sharing values of attributes that are components of the dimensions. The approach will get you to the number of dimensions required for your application, and then you can compare tools according to their capacities to fulfill your requirements.

- **Ability of tool to support analyses against atomic data sets required by the DSS.**

Sometimes ad hoc analyses need to access the basic data underlying the data mart or data warehouse. Does the tool you are considering have this "drill-through" ability or not?

- **Ability of the OLAP tool to integrate directly with relational databases and non-numeric relational data.**

ROLAP tools have a built in advantage in this respect. But individual MDOLAP tools have established connectivity with various relational databases, as has Sybase IQ. You need to know how tools being considered interface with your relational database.

This criterion is a specification of the previous one applied to relational databases; Most OLAP tools will interface with Oracle, Sybase, and Informix, and many have no problem with MS SQL Server. But obviously care must be taken here to see that the OLAP tool supports the version of the database you prefer for the data warehouse.

- **Ability to perform calculations at run-time.**

This is important because the requirements of ad hoc queries are often unanticipated by designers and may not have been directly accomodated in the aggregation or compilation schemes of OLAP tools. The ability of a tool to perform calculations rapidly at run-time is a critical determinant of ad hoc query capability.

- **Data loading performance of the OLAP product.**

This criterion is more critical if the DSS is a frequently updated database. Depending on the business environment, loading can range from a daily affair on up to an annual update. Know what update frequency is required, and calculate the required carrying capacity of the tool accordingly. Match the requirement with benchmark results on the tool being considered.

- **Openness to standard reporting tools.**

In particular, can the product work along with the approved reporting tools in your organization? Or if you're not so constrained can it work with reporting tools you're considering?

- **Ad hoc query performance.**

Benchmarks need to be run to compare products on ad hoc query performance. Benchmarks should test query and reporting tools alone and in combination with OLAP products. Do OLAP products meet requirements for ad hoc query performance? How do they compare to one another when working along with your database?

- **Performance in running standard reports in conjunction with reporting tools.**

Again, benchmarks need to be run on comparative performance, and against any standard specified in the requirements analysis.

- **Training required for the OLAP product.**

Most OLAP products require only light end user training. But how do they compare to each other? And what kind of training is needed for power users and data warehouse/data mart administrative users?

- **Ability to produce full multiuser read-write applications with industrial strength security**.

This comes down to ability of the tool to prototype the DSS application. Is the required security available in the prototype? How does it benchmark?

- **Ability of the tool to integrate with your organization's enterprisewide environment by using standard middleware and client/server communications.**
- **Cost of ownership, training, and installation**.
- **Previous acceptance of the tool as an organizational standard or at least an option compatible with the enterprise computing environment.**

Absent such previous acceptance, selection of one tool over another may involve the often substantial bureacratic costs of a CIO or MIS-based approval process. This may greatly expand the lead time necessary to implement a data mart.

- **Ability to manage a three-tier decision support system in real-time.**

The OLAP tool needs to be able to manage query and report traffic simultaneously with communications with the data warehouse or data mart relational database server, with other analytical server-based data mining tools, with groupware applications, and with the internet.

- **Support in the tool for workflow automation.**

The OLAP tool may need to support a programmed workflow as an important component. Can it be integrated with workflow tools that produce this kind of application?

- **Support for tuning against tool produced performance statistics**.

Does the tool provide statistics that you can use for performance tuning? Without these you probably won't be able to get the performance you need.

- **Extent of analytical capabilities relative to what your DSS application needs, and compared to other tools.**

OLAP tools vary widely in their analytical capabilities. Some produce basic analytical statistics, while others, provide substantial statistical analysis capability. What do you need in your application? What kinds of capabilities will you have outside of the OLAP tool? If these are substantial, how will the OLAP tool interface with your external analytical tools?

- **Support for OLE, and COM or CORBA distributed architecture.**

How distributed will your DSS architecture be? Tools will differ in the degree to which they can make use of distributed computing capabilities. If you're planning to implement a distributed architecture, you'll need an OLAP tool that supports one also.

- **Support for real-time query governing and flow control.**
- **Support for custom application development with standard front - end tools such as Visual Basic, Powerbuilder, and C++.**
- **Tool support for Internet deployment.**
- **Support for Dimension Table designs**

This criterion applies to ROLAP tools. It refers to support in constructing star and snowflake designs, and for queries against star or snowflake dimensional data models.

References

[1] The term OLAP is due to the father of the relational database movement, E. F. Codd. See his "On Line Analytical Processing," Arbor Software White Paper, 1993, for his 12 rules for OLAP.

[2] Microstrategy is already as large or larger than Logic Works, the manufacturer Of ERwin and BPwin.

[3] "OLAP: Scaling to the Masses," Information Advantage, Inc. White Paper, no date.

[4] See Steve Roti, "Riding High on Expressway 103" DBMS (July 1994), 90-92, and Richard Finkelstein, "Sybase IQ: Expressly for the Warehouse" Database Programming and Design (December, 1996), 47-49.

[5] http://www.expertchoice.com

[6] Fatemeh Zahedi, Intelligent Systems for Business: Expert systems with Neural Networks

(Belmont: CA, Wadsworth, 1993). Chapter 12.

[7] Joseph M. Firestone, "A Systems Approach to Dimensional Data Modeling," White Paper No.1, Executive Information Systems, Inc. Wilmington, DE, March 12, 1997 (available from the author).

Biography

Joseph M. Firestone is an independent Information Technology consultant working in the areas of Decision Support (especially Data Marts and Data Mining), Business Process Reengineering and Database Marketing. He formulated and is developing the idea of Market Systems Reengineering (MSR). In addition, he is developing an integrated data mining approach incorporating a fair comparison methodology for evaluating data mining results. You can e-mail Joe at eisai@home.com.

[Up] [Data Warehouses and Data Marts: New Definitions and New Conceptions]
[Is Data Staging Relational: A Comment]
[DKMA and The Data Warehouse Bus Architecture]
[The Corporate Information Factory or the Corporate Knowledge Factory]
[Architectural Evolution in Data Warehousing]
[Dimensional Modeling and E-R Modeling in the Data Warehouse]
[Dimensional Object Modeling] [Evaluating OLAP Alternatives]
[Data Mining and KDD: A Shifting Mosaic]
[Data Warehouses and Data Marts: A Dynamic View]
[A Systems Approach to Dimensional Modeling in Data Marts]
[Object-Oriented Data Warehousing]

Executive Information Systems, Inc.

Data Warehouses and Data Marts: A Dynamic View

By

Joseph M. Firestone, Ph.D.

White Paper No. Three

March 27, 1997

Patterns of Data Mart Development

In the beginning, there were only the *islands of information*: the operational data stores and legacy systems that needed enterprise-wide integration; and *the data warehouse*: the solution to the problem of integration of diverse and often redundant corporate information assets. Data marts were not a part of the vision. Soon though, it was clear that the vision was too sweeping. It is too difficult, too costly, too impolitic, and requires too long a development period, for many organizations to directly implement a data warehouse.

A data mart, on the other hand, is a decision support system incorporating a subset of the enterprise's data focused on specific functions or actvities of the enterprise. Data marts have specific business-related purposes such as measuring the impact of marketing promotions, or measuring and forecasting sales performance, or measuring the impact of new product introductions on company profits, or measuring and forecasting the performance of a new company division. Data Marts are specific business-related software applications.

Data marts may incorporate substantial data, even hundreds of gigabytes, but they contain much less data than would a data warehouse developed for the same company. Also since data marts are focused on relatively specific business purposes, system planning and requirements analysis are much more manageable processes, and consequently design, implementation, testing and installation are all much less costly than for data warehouses.

In brief, data marts can be delivered in a matter of months, and for hundreds of thousands, rather than millions of dollars. That defines them as within the range of divisional or departmental budgets, rather than as projects needing enterprise level funding. And that brings up politics or project justification. Data marts are easier to get through politically for at least three reasons. First, because they cost less, and often don't require digging into organization-level budgets, they are less likely to lead to interdepartmental conflicts. Second, because they are completed quickly, they can quickly produce models of success and corporate

constituencies that will look favorably on data mart applications in general. Third, because they perform specific functions for a division or department that are part of that unit's generally recognized corporate or organizational responsibility, political justification of a data mart is relatively clean. After all, it is self-evident that managers should have the best decision support they can get provided costs are affordable for their business unit, and the technology appears up to the job. Perhaps for the first time in computing history those conditions may exist for DSS applications.

So, data marts have become a popular alternative to data warehouses. As this alternative has gained in popularity, however, at least three different patterns or informal models of data mart development have appeared. The first response to the call for data mart development was the view that data marts are best characterized as subsets (often somewhat or highly aggregated) of the data warehouse, sited on relatively inexpensive computing platforms that are closer to the user, and are periodically updated from the central data warehouse. In this view, the data warehouse is the parent of the data mart.

The second pattern of development denies the data warehouse its place of primacy and sees the data mart as independently derived from the islands of information that predate both data warehouses and data marts. The data mart uses data warehousing techniques of organization and tools. The data mart is structurally a data warehouse. It is just a smaller data warehouse with a specific business function. Moreover, its relation to the data warehouse turns the first pattern of development on its head. Here multiple data marts are parents to the data warehouse, which evolves from them organically.

The third pattern of development attempts to synthesize and remove the conflict inherent in the first two. Here data marts are seen as developing in parallel with the data warehouse. Both develop from islands of information, but data marts don't have to wait for the data warehouse to be implemented. It is enough that each data mart is guided by the enterprise data model developed for the data warehouse, and is developed in a manner consistent with this data model. Then the data marts can be finished quickly, and can be modified later when the enterprise data warehouse is finished.

These three patterns of data mart development have in common a viewpoint that does not explicitly consider the role of user feedback in the development process. Each view assumes that the relationship between data warehouses and data marts is relatively static. The data mart is a subset of the data warehouse, or the data warehouse is an outgrowth of the data marts, or there is parallel development, with the data marts guided by the data warehouse data model, and ultimately superseded by the data warehouse, which provides a final answer to the islands of information problem. Whatever view is taken, the role of users in the dynamics of data warehouse/data mart relationship is not considered. These dynamics are the main subject of this white paper.

To develop this subject the original three models are first developed in a little more detail. This development is followed with a presentation of three alternative models that consider the role of

feedback from users in the development of data warehouses and data marts. Lastly, an analysis of the usefulness of the six patterns of development is given in light of a particular viewpoint on organizational reality.

Development Models Without Explicit User Feedback

The Top Down Model

The top down model is given graphically in Figure One. The data warehouse is developed from the islands of information through application of the extraction, transformation and transportation (ETT) process. The data warehouse integrates

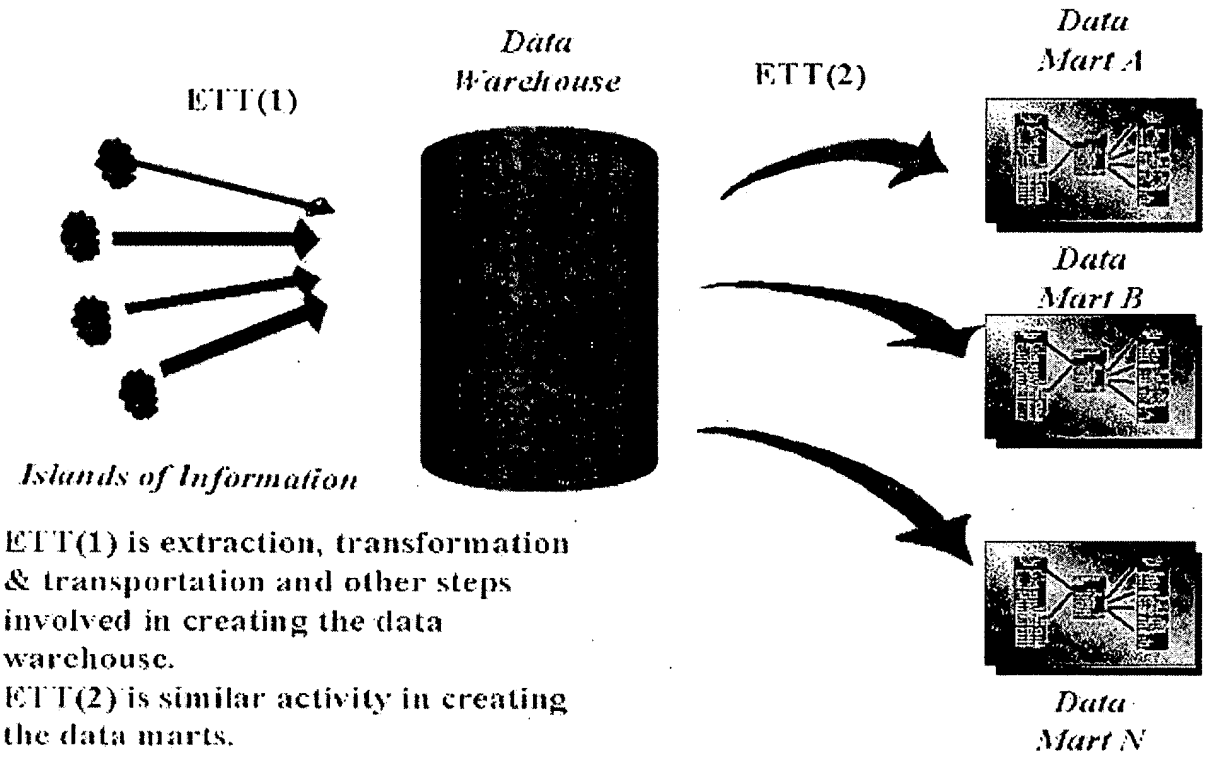


Figure One -- The Top Down Flow from Data Warehouses to Data Marts

all data in a common format and a common software environment. In theory all of an organization's data resources are consolidated in the data warehouse construct. All data necessary for decision support are resident in the data warehouse. After the data warehouse is implemented, there is no further need for consolidation. It only remains to distribute the data to information consumers and to present it so that it does constitute information for them.

The role of the data marts is to present convenient subsets of the data warehouse to consumers having specific functional needs, to help with structuring of the data so that it becomes information, and to provide an interface to front-end reporting and analysis tools that, in turn,

can provide the business intelligence that is the precursor to information. The relation of the data marts to the data warehouse is strictly one-way. The data marts are derived from the data warehouse. What they contain is limited to what the data warehouse contains. The need for information they fulfill is limited to what the data warehouse can fulfill. The data warehouse therefore is required to contain all the data that the enterprise or any part of it might need to perform decision support. And if users discover any need the data warehouse does not meet, the only way to fix the situation is for the users to get the enterprise level managers of the data warehouse to change the warehouse structure and to add or modify the data warehouse as necessary to meet user needs. The model contains no description or explanation of this process of recognition and fulfillment of changing user needs or requirements. But it is inconsistent with the model to assume that data marts would serve as a means of fulfilling changing user needs without changes to the data warehouse occurring first.

The Bottom Up Model

Figure Two depicts the The bottom-up pattern of development. In the left-hand portion of Figure Two, data marts are constructed from pre-existing islands of information, and the data warehouse from the data marts. In this model the data marts are independently designed and implemented, and therefore unrelated to

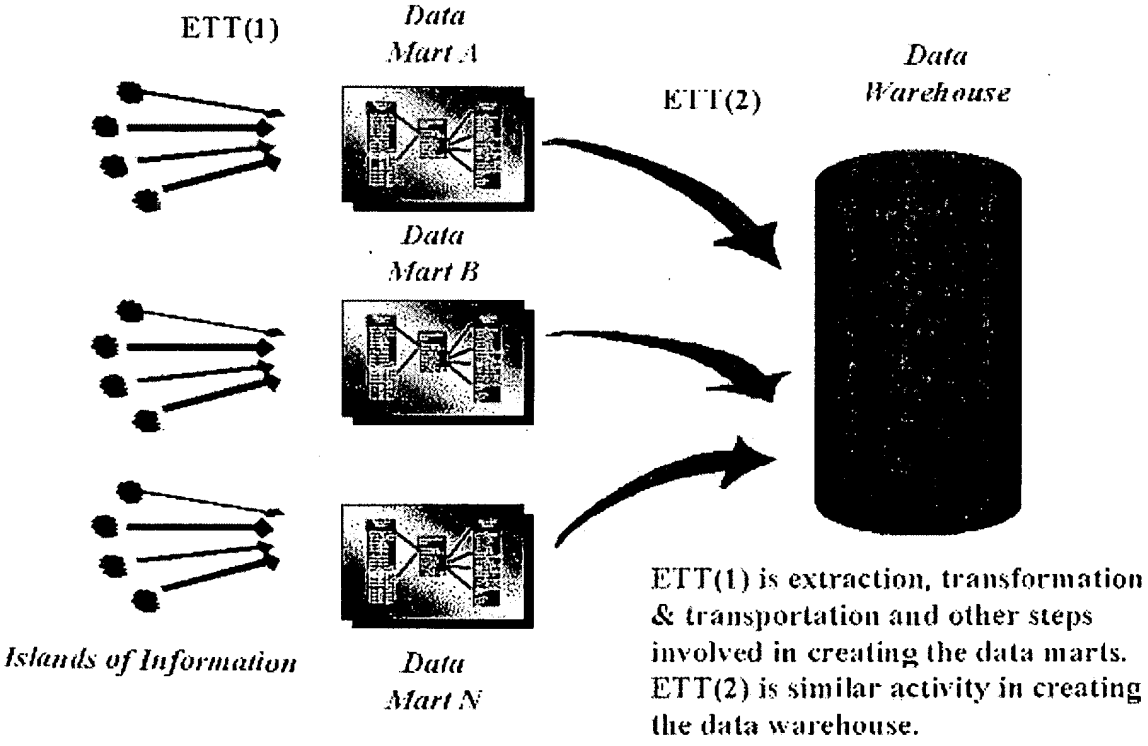


Figure Two – The Bottom-up Flow from Data Marts to the Data Warehouse

one another, at least by design. Growth of this kind is likely to contain both redundancy and important information gaps from an enterprise point of view.

While each data mart achieves an integration of islands of integration in the service of the data mart's function, the integration exists only from the narrow point of view of the business function sponsoring the data mart. From the enterprise point of view, new legacy systems are created by such a process, and these constitute new islands of information. The only progress made is that the new islands employ updated technology. But they are no more integrated and coherent than the old islands were; and they are no more capable of supporting enterprisewide functions.

The right-hand side of Figure Two shows the data mart islands of information being used as the foundation of an integrated data warehouse. A second ETT process supports this integration. It will be needed to remove the redundancy in the data marts, to identify the gaps the process of isolative data mart creation will leave, and to integrate the old islands of information into the new data warehouse in order to fulfill these gaps. The possibility of using older islands of information in this way is not envisioned in this model, which tacitly and incorrectly assumes that the flow from data marts to data warehouse will be adequate to produce a data warehouse with comprehensive coverage of enterprise data needs.

The second model is vague on what happens after the data warehouse is built. Will the data warehouse suddenly become the parent of the data marts, and development proceed according to the top-down pattern? Or will the data warehouse continue to be the "child" of the data marts, which will continue to evolve and lead periodically to an adjustment in the enterprise data warehouse to make it consistent with the changed data marts? The second model doesn't answer such questions, but instead ends its story with the creation of the data warehouse.

Parallel Development

The most popular pattern of development of the first three is the parallel development model. The parallel model sees the independence of the data marts as limited in two ways. First, the data marts must be guided during their development by a data warehouse data model expressing the enterprise point of view. This same data model will be used as the foundation for continuing development of the data warehouse, ensuring that the data marts and the data warehouse will be commensurable, and that information gaps and redundancies will be planned and cataloged as data mart construction goes forward. Data marts will have a good bit of independence during this process. In fact, as data marts evolve, lessons may be learned that will lead to changes in the enterprise data warehouse model. Changes that may benefit other data marts being created, as well as the data warehouse itself.

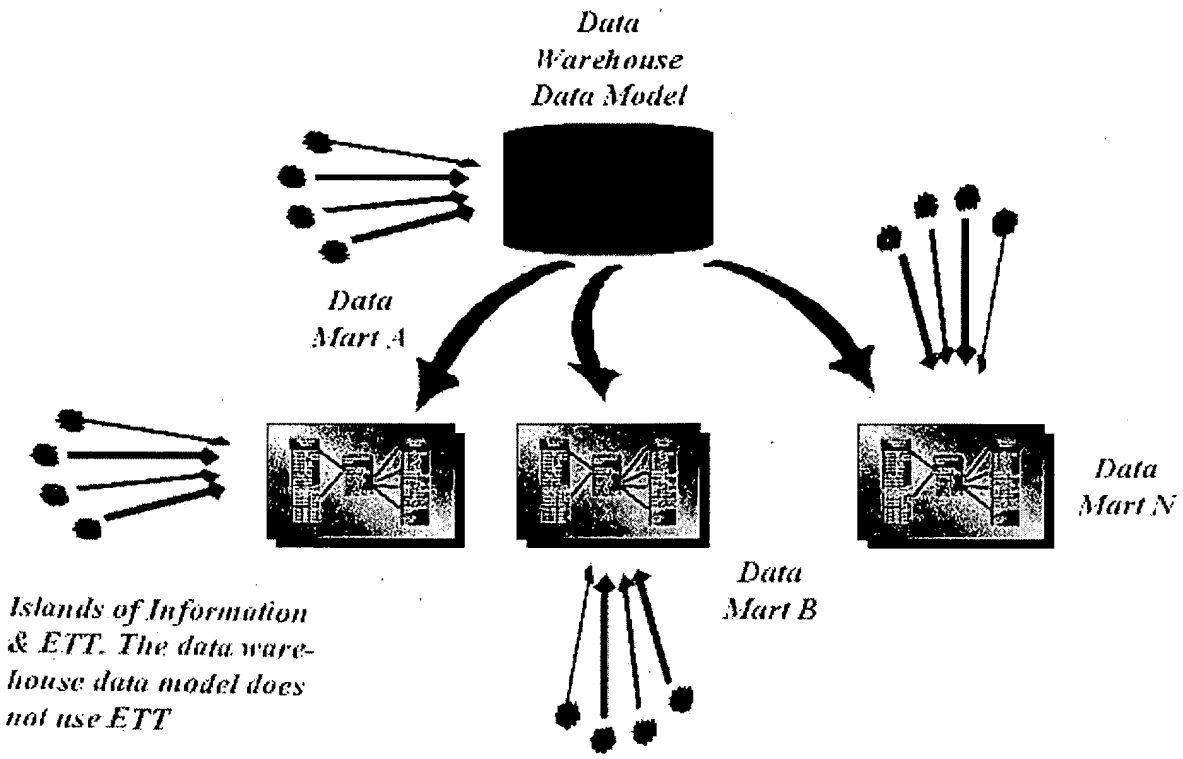


Figure Three -- Data Mart Creation Guided By a Data Model of the Data Warehouse

Second, the independence of data marts is treated as a necessary and temporary expedient on the road to construction of a data warehouse. Once the goal is achieved, the warehouse will supersede the data marts, which will become true subsets of the fully integrated warehouse. From that point on, the data warehouse will feed established data marts, create subsets for new data marts, and, in general determine the course of data mart creation and evolution.

The third pattern begins to treat some of the complexities of the relationship between the data warehouse and data marts. Unlike the first pattern, it recognizes that organizational departments and divisions need decision support in the short-term and will not wait for data warehouse development projects to bear fruit. Thus data marts are necessary and desirable applications for organizations to pursue. Also unlike the first pattern, it sees the data marts as contributing to the data warehouse through evolution in the enterprise data model stimulated by the data marts.

Unlike the second pattern, the parallel view does not provide for uncontrolled growth in data marts. The contents of data marts are to be determined by the enterprise wide data model. Redundancies and information gaps are to be carefully tracked. The enterprise data model will track the activities and accomplishments of data mart projects and be adjusted accordingly. In the parallel development view, data mart activities will contribute to integration of islands of information within the data warehouse, by constituting islands of integrated information within the overall plan provided by the enterprise data model. These islands, in turn, will eventually enter the comprehensive integration of the data warehouse when it is completed.

The third view still retains difficulties. First, it hinges on the rapid development of the enterprise data warehouse model. Decision support consumers will not wait. Not when they have budgets and can support creation of data marts. Though waiting for a data model is a considerably shorter wait than waiting for a full-blown data warehouse, in large organizations the JAD sessions and requirements analyses preceding data model development can take many months. And the job must be done carefully. If the enterprise data model is to guide data mart development, it must be comprehensive in its coverage of data needs. Each time the enterprise model fails to identify a table or attribute necessary for a data mart, a little legitimacy is lost, and the feeling grows that it was not worth waiting for the enterprise data warehouse model, and that it will not be worth waiting for it to be adjusted.

Second, the parallel view, like the other two, also assumes that once the data warehouse is constructed, the data marts will become subsets of the data warehouse, rather than autonomous entities. Parallel development will end, and the data warehouse will fulfill everyone's needs. This assumption is flawed, and envisions a degree of centralization of large enterprises that no longer exists.

The first three patterns of development all fail to explicitly consider continuous user feedback in response to data mart and data warehouse activities. In each view, user requirements are taken into account in constructing data marts or data warehouses, but user requirements are not static and tend to evolve on exposure to new applications and new technologies. Changes in requirements, further, are not limited only to faster hardware, or better techniques for data mining, or improved database software, or GUI interfaces. They may also encompass changes in information and in data requirements that necessitate adding new attributes and tables to data warehouses and data marts, and reorganizing old ones. New requirements may therefore impact data models at both the data mart and data warehouse levels. How will this be handled? Will all new requirements be processed through the data warehouse management? Or will local management implement most new requirements in local data marts first?

Whatever happens will be largely dependent on the nature and amount of feedback from users. The implications of user feedback for the three patterns of development produce three alternative patterns of data warehouse/data mart development. We now turn to these.

Development Models With Feedback

Top Down with Feedback

Suppose your organization is one of the pioneers that implemented a data warehouse before developing any data marts. Suppose the requirements analysis process was done carefully, and the enterprise data warehouse now contains all of the data and conceptual domains suggested or implied by that process. You are now given the assignment to develop an application to carefully measure the performance of your department during the last three years, and to forecast it three months into the future. What does the enterprise data warehouse have to contain to allow you to complete that assignment?

Certainly it has to contain indicators that will track the outcome of performance. Changes in sales, profits, and costs are obvious facts that need to be tracked. But what about *causal* variables, will they be among the attributes of the data warehouse?

The answer is some will be. But unless the effort of creating the data warehouse identified all of the domains within the database, and all of the attributes within those domains by referring to a comprehensive conceptual framework broad enough to encompass concepts and attributes contained in all of the causal models possibly relevant to your department's performance, it is a good bet that the data warehouse will not provide all of the attributes you need. The data warehouse, further, will not be constructed from the causal modeling point of view, unless you or some other representative of your department were considering a data mart at the time the requirements analysis was done for the data warehouse. There would have been no reason for the department's representative to either think in those terms, or to undergo the preparation necessary to think in those terms, in time for the data warehousing JAD sessions, or other requirements gathering tasks.

Your representative would almost certainly have specified essential facts to the data warehouse team, and obvious analytical hierarchies such as: company organization, geography, time, product hierarchies, and so on. But the full makeup of causal dimensions essential for measuring performance, distinguishing it from accident, separating it from either good or bad breaks is likely to be absent.

But you now have the assignment requiring at least some causal modeling, so what do you do? I think you get a subset of the data warehouse for a data mart. But data gathering does not end there. Either you gather data yourself if your department will support that, or you go to external services that sell data relevant for your problem. If you can do either of these two things, you will then supplement the data warehouse subset with the new data, go through a new, if limited, ETT process, and constitute a data mart that will work for your analysis problem.

The assignment your boss gave you has made your requirements change, and that has made the data mart change, and more specifically go beyond the bounds of the corporate data warehouse. You don't want to exceed these bounds, but if you don't, your departmental function suffers, and your job, and your boss's job depend on performing that function, not on maintaining the integrity of the enterprise data warehouse.

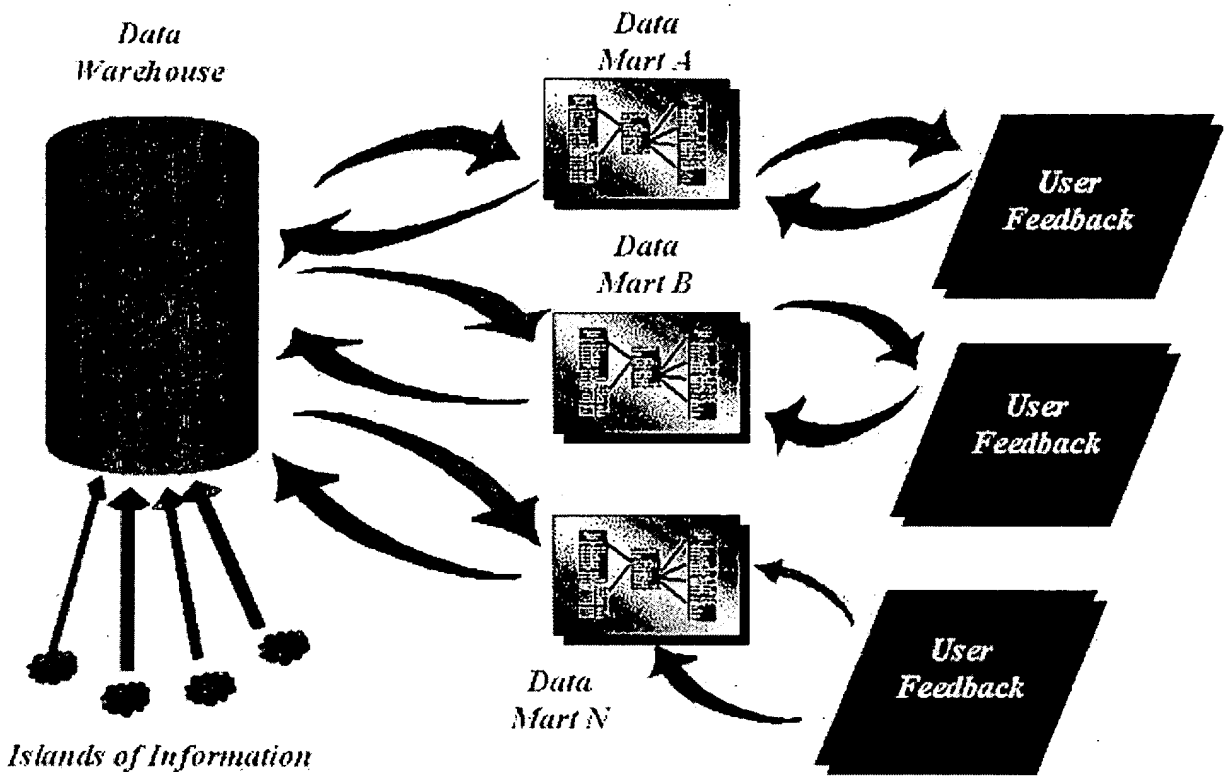


Figure Four -- The Top Down Model with End User Feedback

So this is the first stage of user feedback (See Figure Four). The second stage occurs when the changes made in your data mart are integrated with the enterprise data warehouse. This process can come sooner, or later. It will be sooner, if your company is wise enough to allow continuous feedback from departmental data marts to the data warehouse, and continuous integration of changes that seem necessary at the departmental level. Or alternatively, your company can refuse to recognize the changes being introduced into the data marts at the departmental level. If that's the pattern, the changes could accumulate for years; until there is a new islands of information problem in the company. Then the changes will all come at once with both sides pointing fingers at the other for allowing the data warehouse to get so out of balance with reality.

Whichever pattern applies, the top down model will be subject to departmental user feedback, or adaptation to the top-down data warehouse by departmental data marts. If the continuous pattern of adjustment to departmental changes is adopted, a pattern of gradual evolution of the data warehouses and data marts will occur. The pattern will involve continuous feedback from the periphery to the center, and continual adjustment of both the periphery and the center to each other. The enterprise data warehouse will not bring a once and for-all decision support nirvana, but a much healthier process of continuous conflict and growth in business intelligence.

The Bottom-Up Model With Feedback

The three user feedback models are similar in the possibilities of adjustment to user feedback in the long run. Once the data warehouse is implemented, in each pattern there is the choice of building in a continuous adjustment process between the data warehouse and the data marts, or centralizing further DSS development in the data warehouse (migrating to the top down model). In the short run though, there is considerable difference between the three patterns.

In the top down pattern, user feedback before implementation of the data warehouse is through involvement in the system planning, requirements analysis, system design, prototyping, and system acceptance activities of the software development process. For reasons stated earlier, this involvement is likely to leave gaps in the coverage of domains and attributes that are causal in character, or for that matter that involve unanticipated side effects of departmental performance activities.

In the bottom-up pattern, in contrast, the effect of beginning development with data marts is to ensure much more complete coverage of causal and side effect dimensions. This also means that once the data warehouse is implemented, the bottom-up model with feedback will have little initial gap between user data mart requirements, and what is in the data warehouse. Paradoxically, this small gap could result in an enterprise level decision to migrate to the top down model for long-term development, once the data warehouse is in place. But if this danger is avoided, and the continuous adjustment path to development is followed, then the initial small gap between the data warehouse and data mart requirements will result in a much less painful adjustment process than will be experienced by organizations starting from the top down model. The future should be one of smooth continuous adjustments in the relationships between local data marts and the enterprisewide data warehouse.

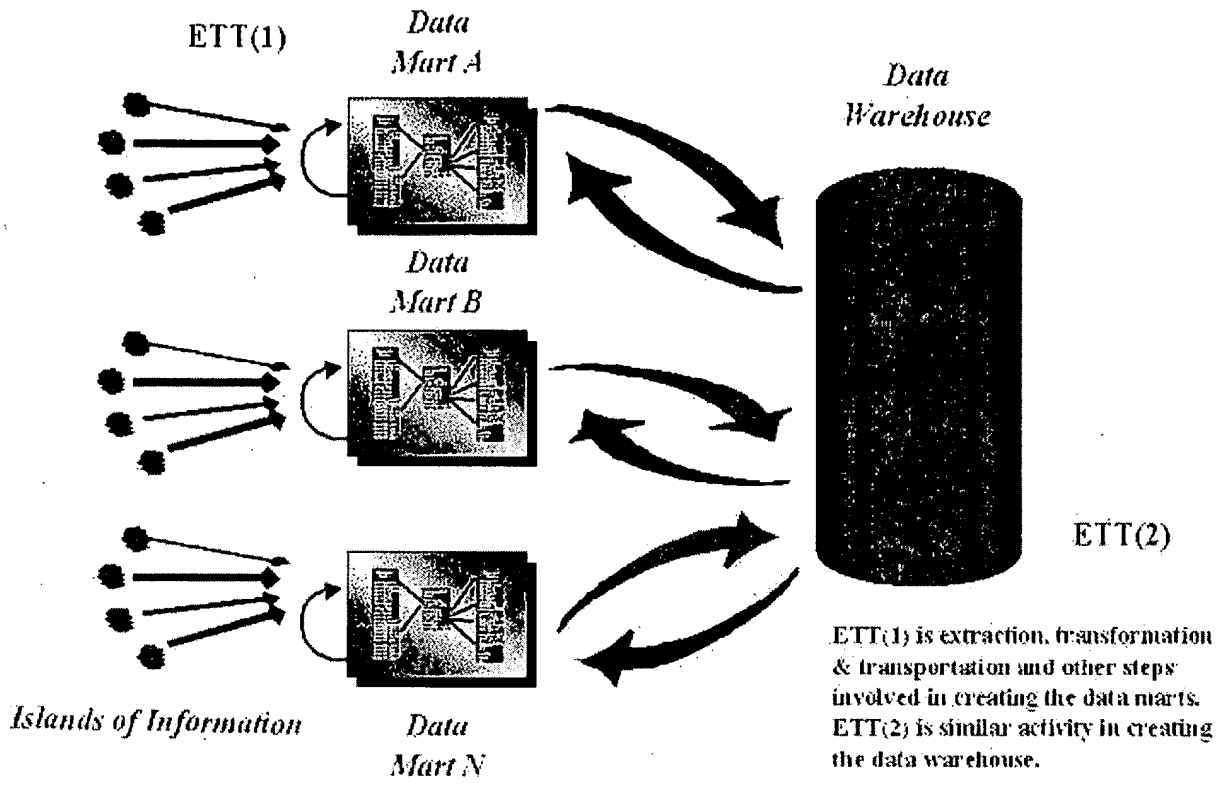


Figure Five -- The Bottom-up Flow from Data Marts to the Data Warehouse with Feedback

Don't conclude though, that the bottom-up model with feedback (See Figure Five) is idyllic. It may not imply much pain once the data warehouse is in place, but if too many data marts are developed for too long in following the bottom-up model, the result is a set of new islands of information, and a painful process of handling redundancies and information gaps in data warehouse construction. So, if the top down model with user feedback means excessive pain in adjusting to data marts following construction of the data warehouse, the bottom-up model can mean excessive pain in integrating information and data from data marts during data warehouse construction.

The Parallel Model With Feedback

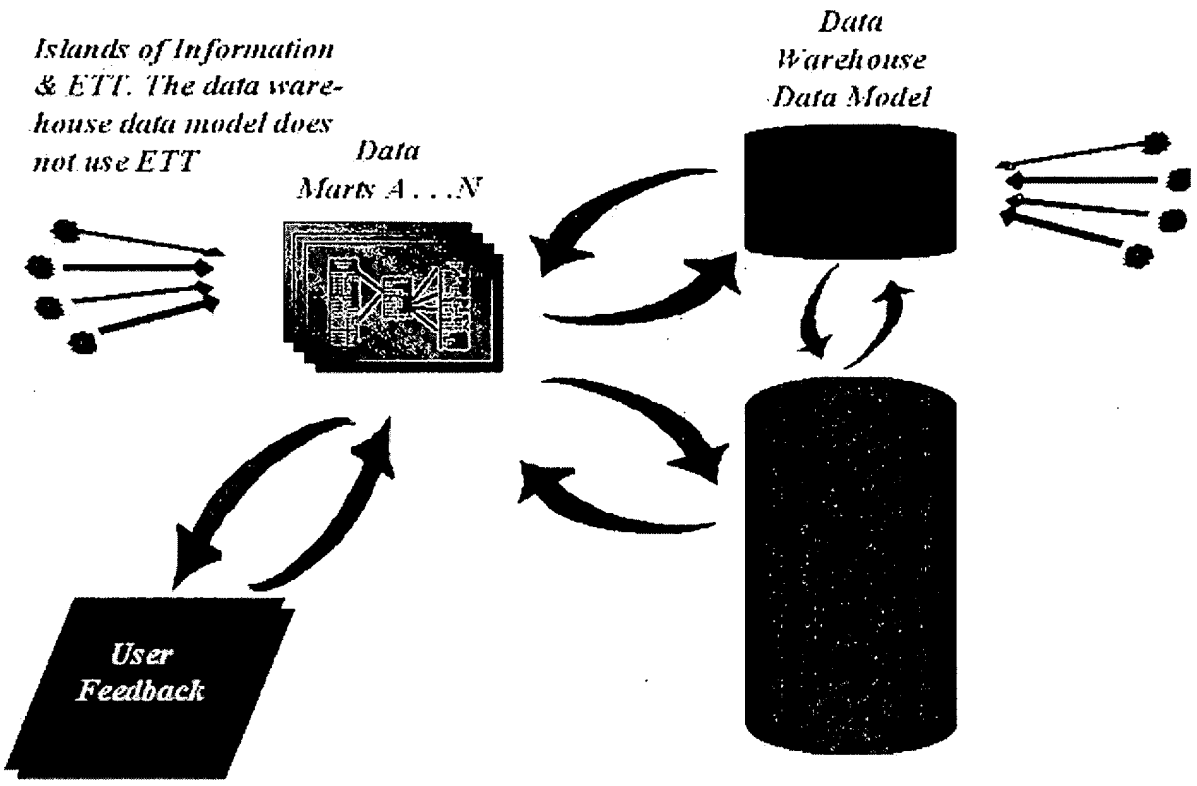


Figure Six – Data Mart Creation Guided By a Data Model of the Data Warehouse with Feedback and An Eventual Data Warehouse

Again, of the three alternative patterns, the parallel model (See Figure Six) offers the most promise. Development begins with a period of mutual adjustment between the enterprise data model and the data marts. As long as the center is open to data mart feedback and adjusts itself to the departmental perspectives on causal and side effect dimensions and attributes, the period of data warehouse development can be relatively smooth. While the data marts should be guided by the enterprise data warehouse model, in a very real sense, the enterprise level model should be guided by the individual and collective input from the data marts. Though the enterprise data warehouse data model is more than the aggregate of collected data mart models, it must certainly encompass those if it is to perform its long-term coordinative/integrative functions.

The danger implementing the parallel model is at the beginning of development. The model assumes completion of the data warehouse data model before data mart development begins, and therefore requires rapid development of the enterprise level model, and also requires the data marts to wait until this development is complete.

This assumption is not necessary for the parallel model. It is probably enough for the data warehouse data model to be in development at the same time as the first data marts, and for the data warehouse to adopt a coordinative and gentle guidance role in common efforts with data mart development staffs. A complete enterprise level data warehouse model is not necessary to monitor and evaluate interdepartmental redundancies, and to track information gaps. Nor is it

necessary to coordinate data mart back-ends to ensure eventual compatibility. On the other hand, if data marts are coordinated by a central modeling team and encouraged to proceed with completion of their data marts with all deliberate speed, their results will inform the enterprise level team of what data warehouse requirements are much more effectively than the most carefully conducted JAD or requirements gathering sessions are likely to do.

The Dynamics of Data Mart Development

The three initial patterns of data mart development are unrealistic in their failure to take account of user feedback to data marts and data warehouses. By introducing explicit consideration of user feedback, one can see that the issue of centralized versus decentralized DSS development is one of long-term as well as short-term significance. All three patterns of development face the key decision of what to do once the data warehouse is developed. Will data marts then be handed down from on high, or will departments and divisions of enterprises have autonomy in evolving their data marts? It is clear that autonomy with central coordination is the most practical course for enterprises in the long run. But the three patterns of development are still distinct choices even if the same long-term policy of mutual adjustment of data marts and data warehouses is followed after data warehouse development.

The top down pattern will require a period of substantial adjustment to data mart needs after the data warehouse is constructed, to moderate centripetal forces and to adjust to the inevitable development of partly autonomous data marts. The bottom-up model will require an extra stage of significant ETT processing to accomodate development of the data warehouse from the data marts. The parallel development model will require rapid development of an enterprise level data warehouse data model unless it is moderated to require only simultaneous development of data marts and the data warehouse, along with coordination from the enterprise team. The parallel development model with feedback and less or no emphasis on a completed data warehouse data model prior to development, seems the indicated "rational" choice for a normative developmental pattern.

But the "rational" choice for development is frequently not a choice that organizations can make. So, an important question is, what will be the distribution of the different patterns of data mart/data warehouse development in organizations? First, none of the first three models will be represented. In neglecting user feedback, they ignore an essential empirical factor in the deverlopment process.

Second, of the alternative patterns, the top down pattern will apply to only a small percentage of enterprises, since it runs counter to the decentralizing forces pervading organizations today. The bottom-up pattern will be popular. Especially if it is supplemented with some coordination from an enterprise-level CIO sponsored data modeling group. Then the worst effects of uncoordinated bottom-up development would be avoided, and the eventual data warehouse would faithfully incorporate the requirements of the data marts.

Finally, the parallel model will also be popular, because it provides for both coordination and

autonomy. It will be still more popular, if it is moderated to require coordination of developing data models, rather than guidance from a completed enterprise level data model. If the bottom-up development pattern is supplemented with coordination from an enterprise level data modeling group, and the parallel model is moderated to abandon the requirement that the enterprise-level data model be completed before beginning data mart development, then the distinction between these two models will blur, and real world cases will only have minor differences in the degree of central coordination of data marts they require. In the end we will see bottom-up and parallel models of data mart development merging, and the final pattern of development will be one of gradual evolution of data marts and data warehouses in a parallel process of mutual adjustment, change, and adaptation to the new problems facing organizations.

Biography

Joseph M. Firestone is an independent Information Technology consultant working in the areas of Decision Support (especially Data Marts and Data Mining), Business Process Reengineering and Database Marketing. He formulated and is developing the idea of Market Systems Reengineering (MSR). In addition, he is developing an integrated data mining approach incorporating a fair comparison methodology for evaluating data mining results. You can e-mail Joe at eisai@home.com.

[Up] [Data Warehouses and Data Marts: New Definitions and New Conceptions]
[Is Data Staging Relational: A Comment]
[DKMA and The Data Warehouse Bus Architecture]
[The Corporate Information Factory or the Corporate Knowledge Factory]
[Architectural Evolution in Data Warehousing]
[Dimensional Modeling and E-R Modeling in the Data Warehouse]
[Dimensional Object Modeling] [Evaluating OLAP Alternatives]
[Data Mining and KDD: A Shifting Mosaic]
[Data Warehouses and Data Marts: A Dynamic View]
[A Systems Approach to Dimensional Modeling in Data Marts]
[Object-Oriented Data Warehousing]

Executive Information Systems, Inc.

A Systems Approach to Dimensional Modeling in Data Marts

By

Joseph M. Firestone, Ph.D.

White Paper No. One

March 12, 1997

OLAP's Purposes And Dimensional Data Modeling

Dimensional Data Modeling (DDM) for Data Marts needs to be viewed from the standpoint of OLAP requirements. Let's take Nigel Pendse's and Richard Creeth's ***F**ast **A**nalysis of **S**hared **M**ultidimensional **I**nformation* (**FASMI**) definition of OLAP [1] as the basis of discussion, since it seems to reflect a greater industry consensus than earlier definitions.

In terms of FASMI, DDM needs to support:

- delivery of most responses to queries in (F) five seconds, with simple queries coming back in less than one second, and all but the very few most difficult queries taking no longer than 20 seconds;
- access to records one needs to perform a variety of analyses (A), including
  - database segmenting (or subsetting according to the criteria specified in a query, also known as "dicing"),
  - rotating (also known as "data slicing,") to examine a different view of the multidimensional data being queried without having to reassemble the view from more basic data,
  - aggregating or disaggregating multidimensional data to display higher or lower levels in an analytic hierarchy such as time periodicity, geography, or business/social/ government organizational hierarchy (known as "rolling up" or drilling down).
  - predictive modeling,
  - time series analysis,
  - measurement modeling combining database attributes (to develop good measures of important abstractions such as corporate or government performance, customer satisfaction, strength of customer bonding, and many other properties not adequately measured by a single database attribute or variable),

- nonlinear, even fuzzy, causal and structural modeling combining measurement and causal models (to develop impact modeling and further refine predictive modeling),
- short- and long-term forecasting,
- automated exploratory data analysis (data mining) to aid Knowledge Discovery in Databases (KDD),
- validation analysis (see my White Paper, "Data Mining and KDD" [2] for an explanation of why validation of data mining is necessary) of patterns discovered through data mining;

- a multidimensional (M) conceptual view of the data in the application;
- a comprehensive organization of all the data (I) that may be needed to achieve KDD.

While current approaches are pretty effective for most of this list, they're not enough for all of it. In particular, not for supporting measurement, causal, and structural modeling, or long-term forecasting; and not for supporting a comprehensive-enough specification and organization of data for KDD. So, I'm moved to propose the following systems approach to dimensional data modeling in data marts. I use the term "systems approach" because it's designed to support development of business area models of system dynamics (in the broadest sense of this term, not to be confused with Jay Forrester's name for his well-known modeling technique, popular in the late 'sixties and the 'seventies). Also, while the statement of this approach is more explicit than his, and is heavier on the prior conceptualization and data inventory side of DDM development, much of it is implicit in Ralph Kimball's brilliant published work. [3]

A Systems Approach to DDM

STEP ONE: Develop a conceptual framework dividing the subject matter to be covered by the data mart into conceptual domains. Each domain should be a category composed of individual concepts, each having two or more possible values. The concepts should be high level abstractions, not data attributes.

Arrive at the domains and their concepts by thinking in terms of the following categories.

(I) Conditions, states, or things that users of the data mart value as goals, or objectives, or criteria related to goals and objectives, of their enterprise or division. These are endogenous to the system being tracked by the data mart, and should be viewed as the effects of certain causes. Favorite examples are Amount of Sales, Unit Costs, Quantity Sold, and Discounts.

When arriving at these concepts don't forget to include possible concepts relating to possible "side effects" only indirectly related to enterprise goals and objectives. Side effects are important because they can lead to feedback effects on key goals and objectives even though they initially may seem unrelated to them.

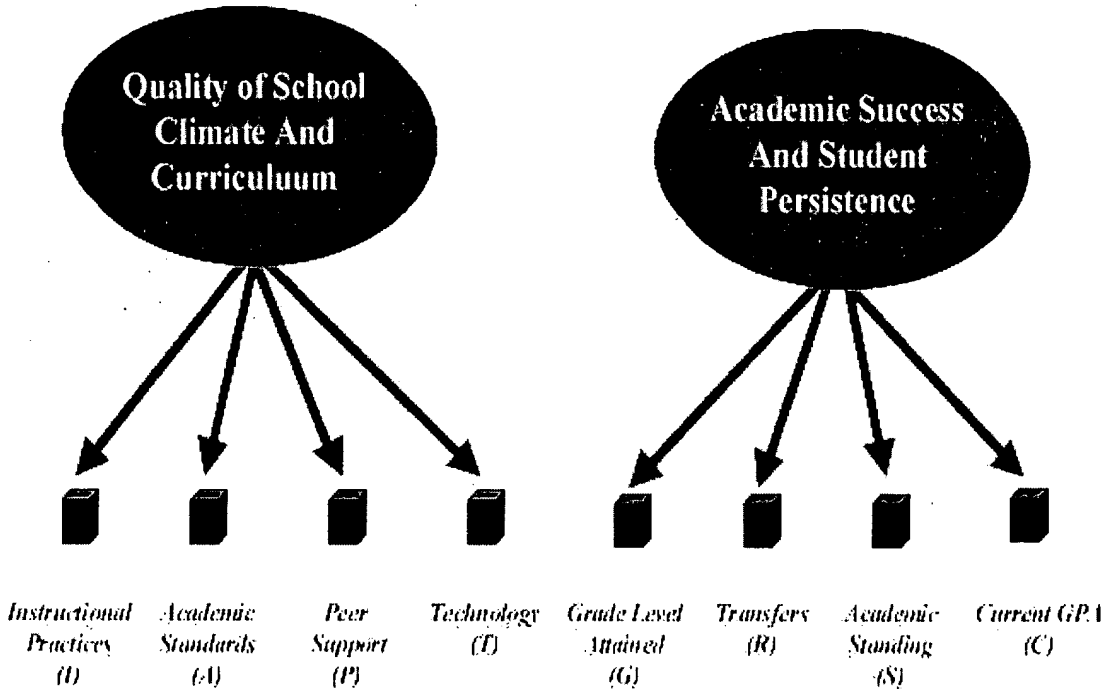


Figure One -- Two Conceptual Domains and Some Associated Concepts

(II) Conditions, states, or things that may conceivably have an impact on the goals or objectives of the enterprise or division developing the data mart. In other words, the possible causes or exogeneous factors such as: advertising or direct mail promotions, interest rates, demographic background of customers, and appearance of technologically sophisticated competitors.

(III) Conditions, states, or things representing descriptive properties that provide a framework for segmenting effects into subtypes useful for understanding the details of effects viewed as business process outcomes. Some examples are product type, and product department. The distinction between Type II causal concepts and Type III descriptive concepts is not hard and fast. It depends on what you think about causality. But in every data mart you'll think of some concepts as causal, and others as primarily descriptive. And your views on these matters may change as the data mart is used and more analysis and data mining is done.

(IV) Components of analytic hierarchies defining levels of description and analysis that provide a framework for either globalizing or localizing descriptions through geographical and social space and/or time. Examples are the time, geographic, organizational, political, product hierarchies whose components are levels of analysis characterized by parent/child or global/local relations.

STEP TWO: Develop the conceptual framework further by specifying abstract cause-effect relations between category (II) causal concepts, and category (I) effects concepts, and by distinguishing the analytic hierarchies for globalizing or localizing measurement. To specify

cause-effect relations you can use a set of formless functional equations that lay out possible or conceivable relationships, but make no comitment to particular linear or non-linear forms. You can go further and formulate more specific causal relations if you have good reasons for doing this, but the data mart DDM need only *provide a framework* for formulating alternative causal models and theories of system dynamics. Specific models can be formulated during subsequent analysis and modeling activities.

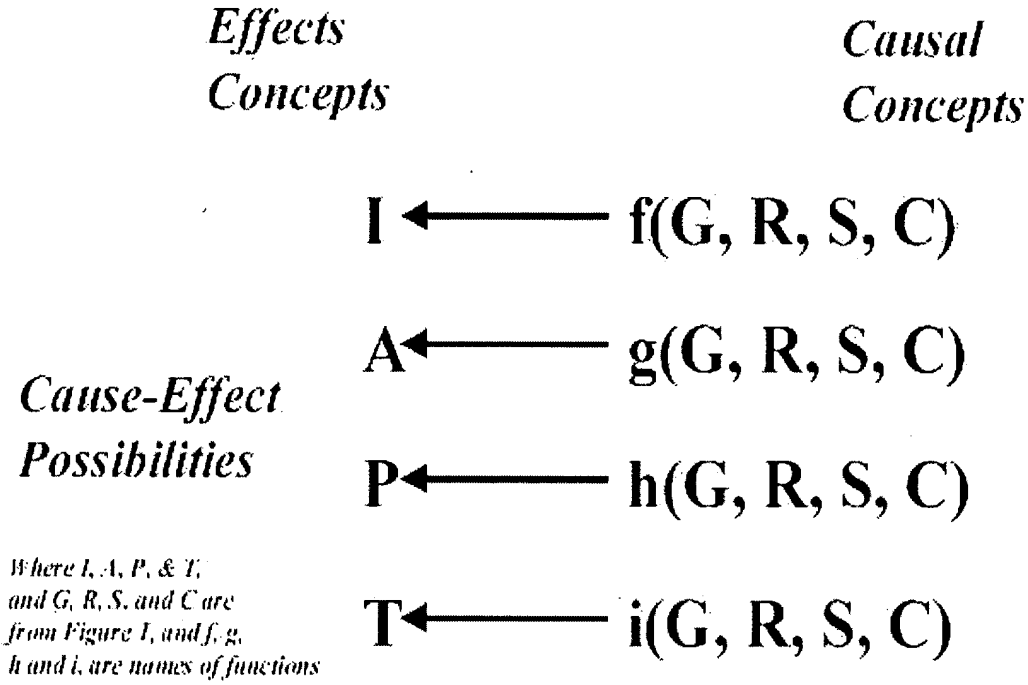


Figure Two -- Specifying Abstract Cause-Effect Relations

STEP THREE: From the initial framework resulting from STEP TWO, extract the conceptual domains and associated concepts you will want to use for the current data mart. Conceptual domains will eventually map to candidate relational tables, though not in general one-to-one, and their associated concepts will guide you to data variables that, in turn, will provide measures of these concepts. Find these data variables by doing a data inventory of all data sources accessible to the data mart team, both internal and external to the sponsoring enterprise. Then map the data variables to the concepts and conceptual domains in order to define candidate attributes for the eventual DDM tables.

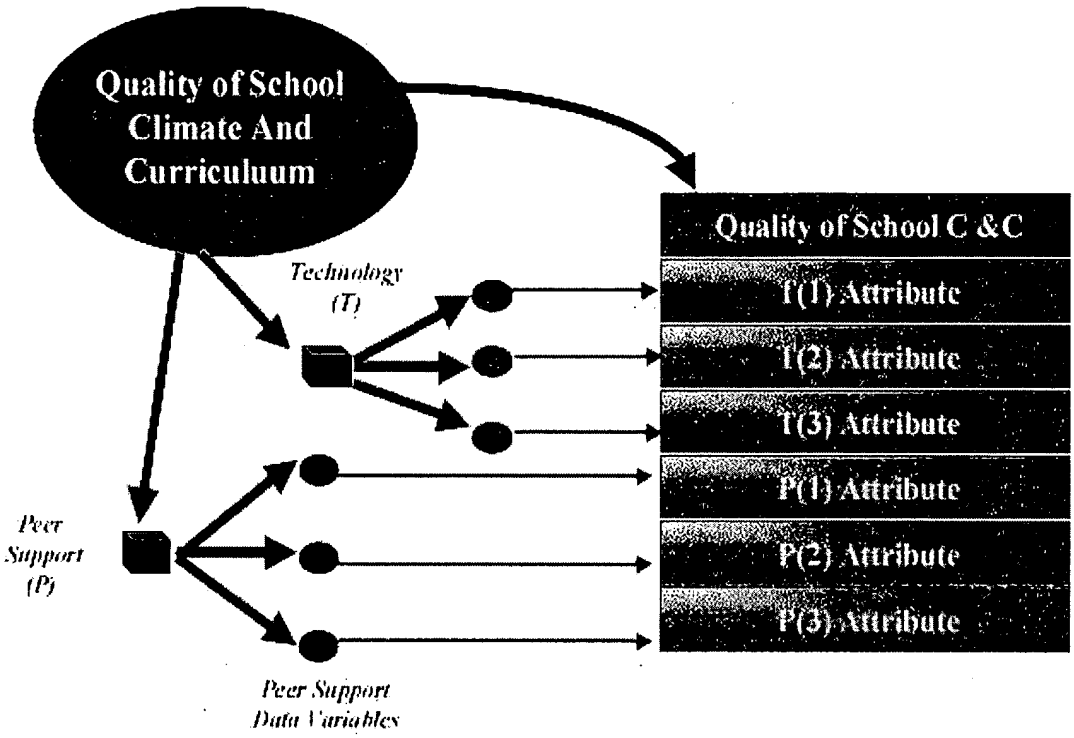


Figure Three -- Mapping a Conceptual Domain to a Table and Data Variables to Attributes

This step provides a foundation in the data mart for later measurement modeling. If the earlier specification of concepts and the data inventory of this step are sufficiently comprehensive, the data mart will have the conceptual and data attribute raw material necessary to provide semantic content to measurement models. While these measurement models will eventually become part of the data mart, they are not part of the dimensional data model. It specifies only the Tables and attributes for the measurement models to operate on.

STEP FOUR: Find the conceptual domains (candidate tables) that contain the candidate attributes measuring the presence, absence or closeness of approach to goals and objectives. Pick these domains as sources of attributes for candidate Fact Tables. Think about the attributes you need to model in this data mart and decide on the meaning of the lowest level record in the data mart containing these attribute(s). Following Kimball, this is called deciding on the grain of the Fact Table.

Now, identify the attribute(s) that will serve as the grain attribute(s) for the Fact Table(s) of the data mart. The grain attributes of the Fact Tables provide data useful in measuring the effects of causal attributes on the goals and objectives of the enterprise. It is this data, the changes in it, and the measures derived from it, that are an important focus of tracking in the data mart.

STEP FIVE: Identify the conceptual domains whose attributes may have a causal impact on the Fact Table grain attribute(s). Map these attributes to relational tables bearing the names of the conceptual domains the attributes are intended to measure or describe. Specify primary keys for

these tables.

STEP SIX: Identify the conceptual domains whose attributes can provide more detailed descriptions of the Fact Table grain attribute(s). Map these attributes to relational tables bearing the names of the conceptual domains they are intended to measure or describe. Specify primary keys for these tables.

Sometimes causal and descriptive attributes will share a conceptual domain and will map to the same relational tables. Sometimes mapping will result in tables that have only descriptive or causal attributes in them apart from keys. Depending on results it may be possible to clearly distinguish certain dimensions as causal, others as descriptive, and others as combinations of both.

Kimball's promotion dimension is distinctly causal. All the attributes in it are attributes measuring exogenous manipulative variables, under control of the enterprise, that can impact sales. His treatment suggests that when a dimension represents enterprise policies, strategies, or tactics, it can always be considered a causal dimension. There are some purely causal dimensions though, that the enterprise may not be able to manupulate. For example, dimensions representing external economic conditions. Or dimensions representing stock market cycles. In addition, though other dimensions such as product or personnel dimensions may not be purely causal, there will be some attributes measuring causal factors in these dimensions.

STEP SEVEN: Identify the conceptual domains that lay out analytic hierarchies for localizing or generalizing the description provided by the Fact Table grain, including description and measurement of change over time. Decide whether to incorporate these domains as separate Hierarchy Tables, or as fields in other previously defined tables. Specify their primary keys.

A time dimension table is almost always included in data marts. But frequently, other hierarchies such as geography, sales organization, product type, etc., are included in the same tables as non-hierarchical factors. How the tables are formulated is guided by performance considerations.

STEP EIGHT: Having specified the keys in all dimensional tables, go back to the candidate Fact Tables and add the primary keys of Causal, Descriptive, Hierarchy, and combined dimensions as foreign keys. Consider, next, whether the Fact Tables need to be supplemented with Factless Event or Coverage Tables. In general, Factless Tables will be needed to measure the impact of hypothesized causes on Fact Table effects, because these tables measure the absence of effects. Create the Factless tables by deciding whether event, coverage, or both types of tables are necessary in your design. Include the same foreign keys in the Factless tables as are in the corresponding Fact Tables.

STEP NINE: Use the hierarchical and combined dimensions of the DDM to define logic for computing aggregative tables from the Fact, Factless, Causal, Descriptive, and combined dimensions.

DDM Benefits

This systems approach (Figure 4) requires highly explicit, top-down conceptualization and data inventory steps in order to sketch out the broadest possible view of the outlines of the range of measurement and cause-effect relations underlying the data mart. The conceptualization and data inventory activities establish the semantic content of the DDM. They provide the foundation necessary for future efforts at KDD to use the concepts and attributes underlying the data mart to successfully conclude a wide range of measurement, causal, structural, time-series, forecasting, and dynamic analyses including explicit modeling, data mining and data validation.

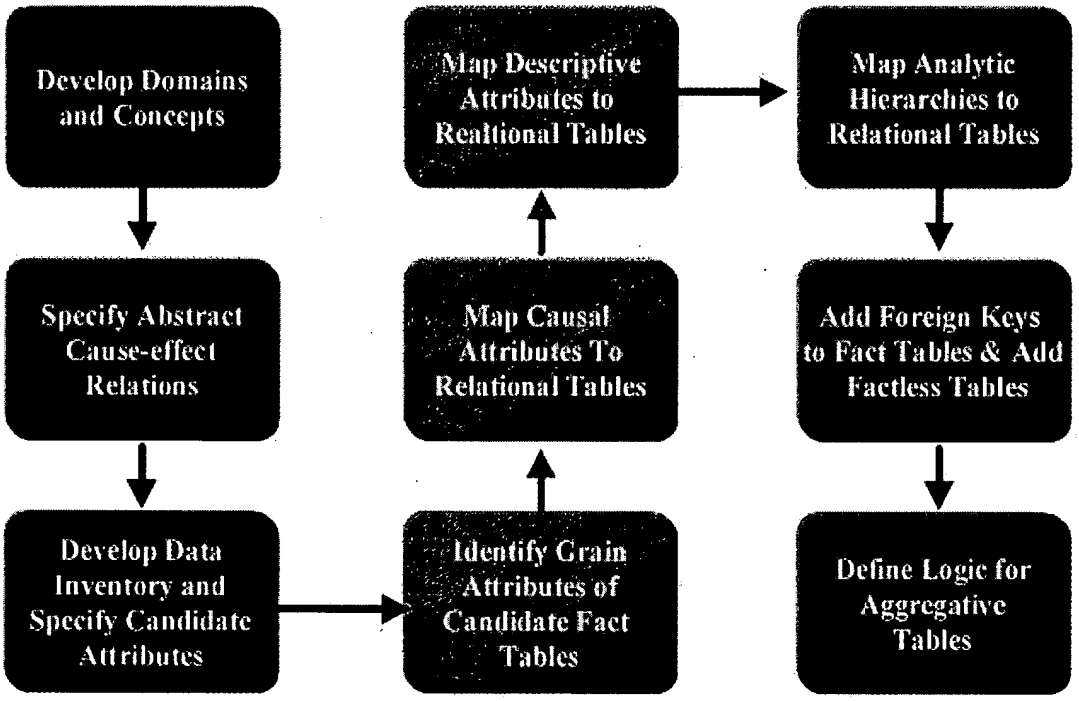


Figure 4 -- The Nine Steps of the Systems Approach to DDM

In following this approach the data modeler also justifies selection of tables and attributes because of their actual and possible connections to the business goals and objectives of the enterprise, and the low level criteria represented by the Fact Table attributes related to these goals and objectives. In addition, the approach follows Kimball's work closely enough to ensure that all the other aspects of OLAP as defined in the FASMI definition are also supported.

References

[1] Nigel Pendse and Richard Creeth, "Synopsis of the OLAP Report," Business Intelligence, Inc., Norwalk, CT, 1997 (available at http://www.busintel.com/synopsis.htm).

[2] Joseph M. Firestone, "Data Mining and KDD: A Shifting Mosaic," White Paper No.2, Executive Information Systems, Inc. Wilmington, DE, March 12, 1997 (available from the author).

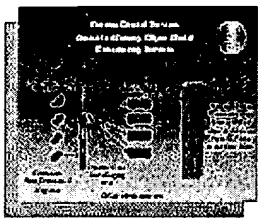
[3] Ralph Kimball, The Data Warehouse Toolkit. New York: N. Y. (John Wiley and Sons) 1996.

Biography

Joseph M. Firestone is an independent Information Technology consultant working in the areas of Decision Support (especially Data Marts and Data Mining), Business Process Reengineering and Database Marketing. He formulated and is developing the idea of Market Systems Reengineering (MSR). In addition, he is developing an integrated data mining approach incorporating a fair comparison methodology for evaluating data mining results. You can e-mail Joe at eisai@moon.jic.com.

[Up] [Data Warehouses and Data Marts: New Definitions and New Conceptions]
[Is Data Staging Relational: A Comment]
[DKMA and The Data Warehouse Bus Architecture]
[The Corporate Information Factory or the Corporate Knowledge Factory]
[Architectural Evolution in Data Warehousing]
[Dimensional Modeling and E-R Modeling in the Data Warehouse]
[Dimensional Object Modeling] [Evaluating OLAP Alternatives]
[Data Mining and KDD: A Shifting Mosaic]
[Data Warehouses and Data Marts: A Dynamic View]
[A Systems Approach to Dimensional Modeling in Data Marts]
[Object-Oriented Data Warehousing]

EIS



DKMS Briefs

Joseph M. Firestone

DKMS Brief No. Five: Is Data Staging Relational? A Comment

Introduction

In the data warehousing process, the data staging area is composed of the data staging server application and the data store archive (repository) of the results of extraction, transformation and loading activity. The data staging application server temporarily stores and transforms data extracted from OLTP data sources and the archival repository stores cleaned, transformed records and attributes for later loading into data marts and data warehouses.

A recent question raised by Ralph Kimball [1] is whether the data staging area is relational or has more to do with sequential processing of flat files. He concludes that "most data staging activities are not relational, but rather they are sequential processing. If your incoming data is in flat-file format you should finish your data staging processes as flat files before loading it into a relational database." [1, P. 71] He also states that if both the source and target databases are relational it may be appropriate to retain this format and not convert to flat files.

This answer to the question of the character of the data staging area assumes that the issue boils down to the nature of the file processing associated with the database used in the data staging database servers. But I think the issue is broader than this, and encompasses both the database format and process logic characteristic of: the data staging application, the archival repository, and the metadata and associated metamodel driving the data staging process. This brief examines the above issue. It briefly describes the data staging process, and then discusses the nature of the data staging application, repository, and the metadata and metamodel that drive the data staging process.

The Data Staging Process

The data staging process imports data either as streams or files, transforms it, produces integrated, cleaned data and stages it for loading into data warehouses, data marts, or Operational Data Stores. Kimball, [2] and Kimball, Reeves, Ross and Thornthwaite [3] provide clear detailed accounts of the specific work that is performed in data staging. There is no need to repeat the details of their descriptions. But I would like to highlight a few points relevant to the later discussion.

First, Kimball et.al., distinguish two data staging scenarios. In (1) a data staging tool is available, and the data is already in a database. The data flow is set up so that it comes out of the source system, moves through the transformation engine, and into a staging database. The flow is illustrated in Figure One.

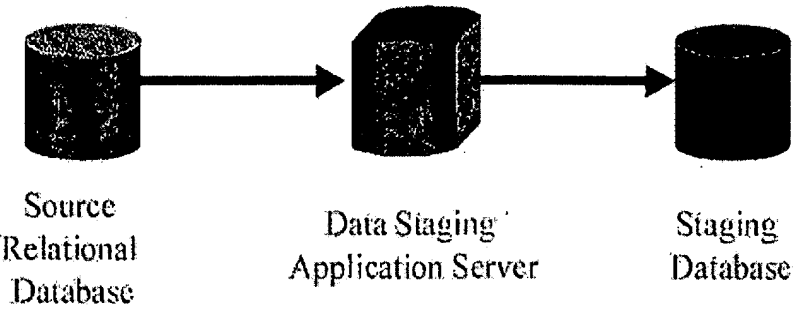


Figure One -- First Data Staging Scenario

In the second scenario, begin with a mainframe legacy system. Then extract the sought after data into a flat file, move the file to a staging server, transform its contents, and load transformed data into the staging database. Figure Two illustrates this scenario.

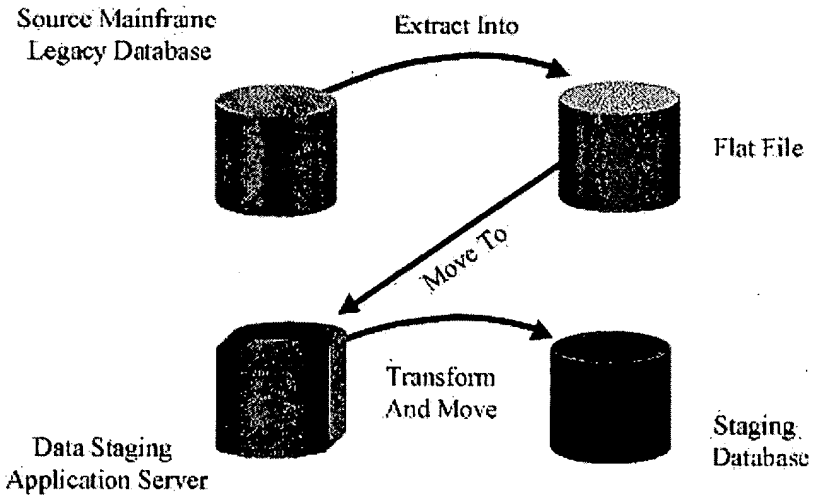


Figure Two -- Second Data Staging Scenario

Second, "almost all processing in the data staging process is sorting, followed by a single sequential pass through one or two tables." [3, P. 616]. This suggests that a relational data staging server is not desirable, in general, though it may be advantageous if both the input data source and the eventual target for loading processed data is an E-R structured database. Otherwise, flat files may be the preferable physical format for the batch sequential processing data staging server application, and perhaps for the data staging database as well.

Third, "the data staging area will archive and store data for a number of purposes. Conformed dimensions are created in the data staging area and replicated out to all the requesting data marts. These conformed dimensions must be permanently housed in the data staging areas as flat files ready for export. The data staging area may be the best place to hold data for emergency recovery operations, especially if the data mart machines

are remote affairs under the control of user departments. The data staging area must also be the source of the most atomic transactional data, especially if the client data marts do not use all of that data at any given point in time. This atomic data then becomes available for further extraction. Again, this archival data may well be stored as flat files available for export or processing by a variety of tools." [3, P. 345]

Fourth, the data staging process is driven in an essential way by metadata, including business rules. Metadata is used along with administrative tools to guide data extractions, transformations, archiving, and loading to target data mart and data warehouse schemas. What is the nature of this metadata? Should it be relational in character? Is it handled best through flat files? Or is there yet another alternative?

Database Type and the Data Staging Application Server

What format should the data be in prior to transformation? If the required data is already in a relational database, use that database for the data staging application server. Don't take resources to transform into flat files, especially if the target presentation database is relational, and is based on the same product. But relational structure is not as efficient for transforming data as a flat file structure, so there will be a performance penalty, which hopefully will be compensated for by the ease of loading into the target relational database.

If the required data is in a flat file, keep it in the flat file. Data staging and cleansing tools have sophisticated batch sequential processing capabilities for sorting and merging flat files; and since almost all processing in data staging is sorting and merging, it wastes time and resources to convert to any other format prior to transformation.

Database Type and the Data Staging Repository

What format should the data be in following transformation and prior to loading? If the data was in relational format prior to transformation processing, then keep it in that format for immediate loading into a presentation relational or multidimensional database server. If dimension and fact tables are to be archived for future data marts, then output these in flat file format, as this is most convenient for export to a variety of tools and applications.

If the data was in flat file format before processing, keep it in flat file format for immediate loading into a relational database. Or if you want to archive it, keep it in flat file format for later export to various applications or to data marts.

Metadata, Metamodel, and Data Staging

The data staging process is metadata and metamodel driven. The metadata is currently expressed in relational format, and the associated metamodel that uses metadata to drive data staging, is expressed as procedural code, frequently scripted in a language provided by a tool vendor.

Both Metadata and its associated metamodel could be expressed in an object model, where metadata would be encapsulated as object attributes, and the rules defining the metamodel could be encapsulated as object methods. This is the trend in data staging process development.

It is reflected in the appearance of such tools as Template's EIT, [4] and Daman Consulting's InfoManager. [5] It is reflected in Informatica' s [6] commitment to develop it's MDX2 as a DCOM-compatible object model. It is also reflected in the development of Distributed Knowledge Management Architecture, [7] the architecture of Distributed Knowledge Management Systems (DKMS). [8]

DKM architecture may be viewed as adding an object layer to architectures based on relational constructs and

logics. [9] The object layer is added to provide integration through automated change capture and management. Figure Three depicts DKM architecture in a data warehousing context from the viewpoint of the role of the object layer.

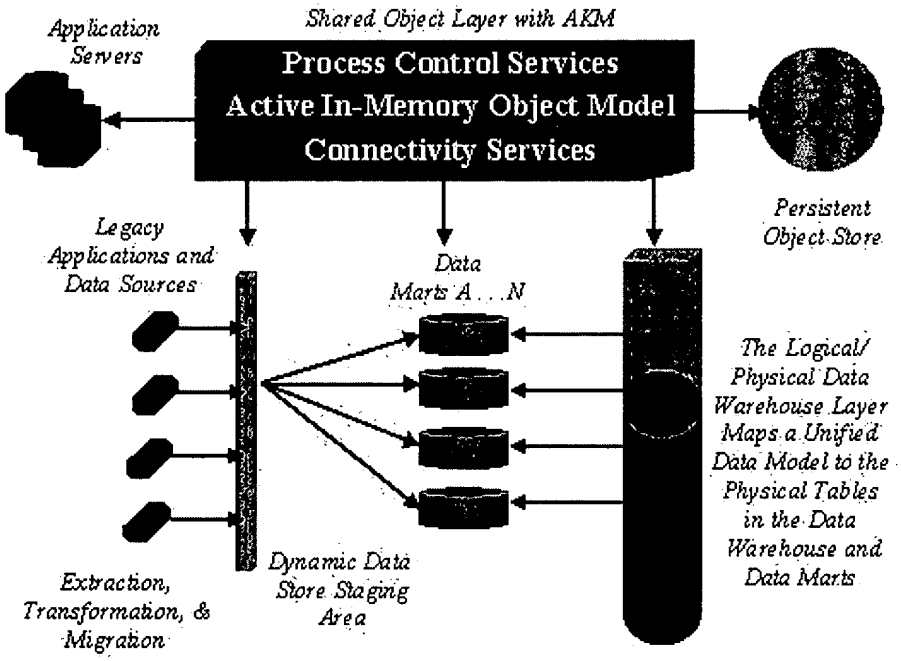


Figure Three -- DKM Architecture

In Figure Three the object layer unites and integrates all DKMS components. The object layer requires an architectural component called an Active Knowledge Manager (AKM) [10]. An AKM provides process control services, an object model of the DKMS, and connectivity to all enterprise information, data stores, and applications. The AKM's object model includes entity objects encapsulating metadata and control objects encapsulating related business process rules (metamodels). The preferred format for persistent metadata and metamodel storage is the format of an Object-Oriented Database Management System (OODBMS). The OODBMS form is not necessary for data staging metadata and metamodels, since the AKM can access persistent metadata in a variety of formats. But it is the only form that avoids the performance penalty caused by the "impedance mismatch" between the AKM and relational, flat file, hierarchical or other non-O-O forms of data storage.

Conclusion

The data staging area is not simply relational, but it is also not simply sequential/flat file in character. Table One provides a summary of the variation in preferred storage format across different aspects of the data staging area.

Table One -- The Data Staging Area and preferred Database Type

Data Staging Component	Data Base Type		
	Flat File	Relational	OODBMS
Data Staging Application Server	If already in Flat File	If already Relational	Waste of Resources
Data Staging Repository	If already in flat file or if dimension and fact tables are to be archived	If already Relational	Waste of Resources
Metadata and Metamodel Repository	----	-----	Preferred Format Is OODBMS

The picture offered in Table One is, more detailed than previous descriptions of the problem, and makes the point that data staging has multiple aspects and not just file processing. It shows appreciable roles for all three types of storage somewhere in the data staging process. Flat files and relational databases are dominant, but there is a definite role for OODBMSs, and it is one that will probably increase along with the commitment to metadata in data warehousing.

References

[1] Ralph Kimball, "Is Data Staging Relational?" DBMS, April 1998, Pp. 14, 16, 71

[2] Ralph Kimball, The Data Warehouse Toolkit (New York: John Wiley, 1996).

[3] Ralph Kimball, Laura Reeves, Margy Ross, and Warren Thornthwaite, The Data Warehouse Life Cycle Toolkit (New York: John Wiley & Sons, 1998)

[4] Template Software, "Integration Solutions for the Real-Time Enterprise: EIT - Enterprise Integration Template," Dulles, VA, White Paper May 8

[5] Inquire at: http://www.damanconsulting.com

[6] Informatica's "second generation" Metadata exchange Architecture includes a commitment to COM and Object Technology. According to Informatica: "MX2 includes a powerful, object-oriented framework based on Microsoft's COM technology, making it interoperable with other COM-based programs and repositories, including the Microsoft Repository. MX2 will comply with the Unified Modeling Language (UML) an Object Management Group standard currently supported by Microsoft, Informatica and other industry leading IT companies."

[7] "Architectural Evolution in Data Warehousing." available at http://www.dkms.com/White_Papers.htm.

[8] I introduced the DKMS concept in two previous White Papers "Object-Oriented Data Warehouse," and "Distributed Knowledge Management Systems: The Next Wave in DSS." Both are available at

http://www.dkms.com/White_Papers.htm.

[9] "Architectural Evolution . . ." P. 12-14

[10] Ibid.

Biography

Joseph M. Firestone is an independent Information Technology consultant working in the areas of Decision Support (especially Data Marts and Data Mining), Business Process Reengineering and Database Marketing. He formulated and is developing the idea of Market Systems Reengineering (MSR). In addition, he is developing an integrated data mining approach incorporating a fair comparison methodology for evaluating data mining results. Finally, he is formulating the concept of Distributed Knowledge Management Systems (DKMS) as an organizing framework for the next business "killer app." You can e-mail Joe at eisai@home.com.

[Up] [Data Warehouses and Data Marts: New Definitions and New Conceptions]
[Is Data Staging Relational: A Comment]
[DKMA and The Data Warehouse Bus Architecture]
[The Corporate Information Factory or the Corporate Knowledge Factory]
[Architectural Evolution in Data Warehousing]
[Dimensional Modeling and E-R Modeling in the Data Warehouse]
[Dimensional Object Modeling] [Evaluating OLAP Alternatives]
[Data Mining and KDD: A Shifting Mosaic]
[Data Warehouses and Data Marts: A Dynamic View]
[A Systems Approach to Dimensional Modeling in Data Marts]
[Object-Oriented Data Warehousing]